

Web Services and REST

This content is protected and may not be shared, uploaded, or distributed.

Introduction

- Web sites are normally accessed by a browser guided by a person
- But we have seen that **programs** can also access a web site, return one or more pages, and scrape the site for information
- Web Services is the idea of offering the capabilities/information on a web site via a programming interface, so application programs can more readily access the information on the site
- *Web Services* are APIs for accessing a website's information across the Internet

Introduction (cont' d)

- The implementation of Web Services is roughly divided into three categories:
 - Big Web Services which involve XML messages that are communicated by the Simple Object Access Protocol (SOAP); the API is formally described using the Web Services Description Language (WSDL). These services are normally used for server-to-server communication, using additional protocols like XML Security and XML Encryption.
 - REST (Representational State Transfer) Services which use HTTP methods PUT, GET, POST and DELETE.
 - Cloud Services which provide cloud storage, application hosting, content delivery, and other hosting services.
- All three types of Web Services provide access through APIs.
- The rest of the slides will cover **REST** Services and **Cloud** Services

REST Services

- Many web sites are now offering their facilities through REST Web Services
- REST Services can be used to access sites that perform the following functions:
 - Web Search (e.g., Google Custom Search)
 - Geolocation (e.g., Google Maps Geolocation API)
 - Photo Sharing (e.g., SmugMug's Flickr)
 - Social Networking (e.g., Facebook, Twitter)
 - Mapping (e.g., Google Maps, Bing Maps)
- Access is provided using one or both of these methods:
 - **Direct URL**, returning a response in one or more formats (XML, JSON, PHP)
 - **Library-based APIs**, embedded in JavaScript, Java, C#, Objective-C and other source and binary library formats
- Many of these services now require or include **OAuth user authentication**
 - OAuth is a standard for clients to access server resources on behalf of a resource owner
 - E.g., see <http://en.wikipedia.org/wiki/OAuth>
- Many of these services limit daily usage by a single website, and require payment when the thresholds are breached

Cloud Services

- Cloud Services covers a variety of hosting services:
 - Application Hosting (e.g., AWS, Google App Engine, FireHost, Microsoft Azure)
 - Backup and Storage (e.g., AWS)
 - Content Delivery (e.g., Netflix hosted by AWS)
 - E-commerce (Amazon.com e-commerce)
 - Media Hosting (e.g., Microsoft Azure, RackSpace, Streaming Media Hosting)
 - DNS Protection Services (e.g., CloudFlare)
 - Consumer Cloud Storage (e.g., Apple iCloud Drive, Dropbox, Microsoft OneDrive, Google Drive)
- Access is provided using one or both of these methods:
 - Dashboard
 - Library-based APIs, embedded in Java, C#, Objective-C and other binary library formats
- All these services are commercial services that require monthly payments
- The consumer cloud services provide limited, free basic storage

REST (Representational State Transfer)

- REST is a style of software architecture for distributed hypermedia systems
 - Initially proposed by **Roy Fielding** in a 2000 doctoral dissertation (remember which RFC he was involved with? The HTTP specification.) He co-founded the Apache HTTP Project.
 - See: <http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm>
 - The World Wide Web is an example of REST
- There are three fundamental aspects of the REST Design Pattern
 - 1. **client**, 2. **servers** and 3. **resources**
 - Resources are typically represented as documents
 - Systems that follow Fielding's REST principles are often referred to as **RESTful**;

Resources

Every distinguishable entity is a resource

URLs

Every resource is uniquely identified by a URL



Simple Operations
(PUT,GET,POST,DELETE)

REST versus Other Approaches

- **REST**

- Software architectural style for distributed hypermedia systems like WWW
- Quickly gained popularity through its **simplicity**

- **SOAP**

- Protocol for exchanging XML-based message, normally using HTTP
- Much more robust way to make requests, but more robust than most APIs need
- More **complicated** to use
 - https://www.w3schools.com/xml/xml_soap.asp

- **XML-RPC**

- RPC protocol with XML as an encoding and HTTP as a transport
- More complex than REST but much simpler than SOAP
- Supported by Python:
 - <https://docs.python.org/3/library/xmlrpc.html>

- **JSON-RPC**

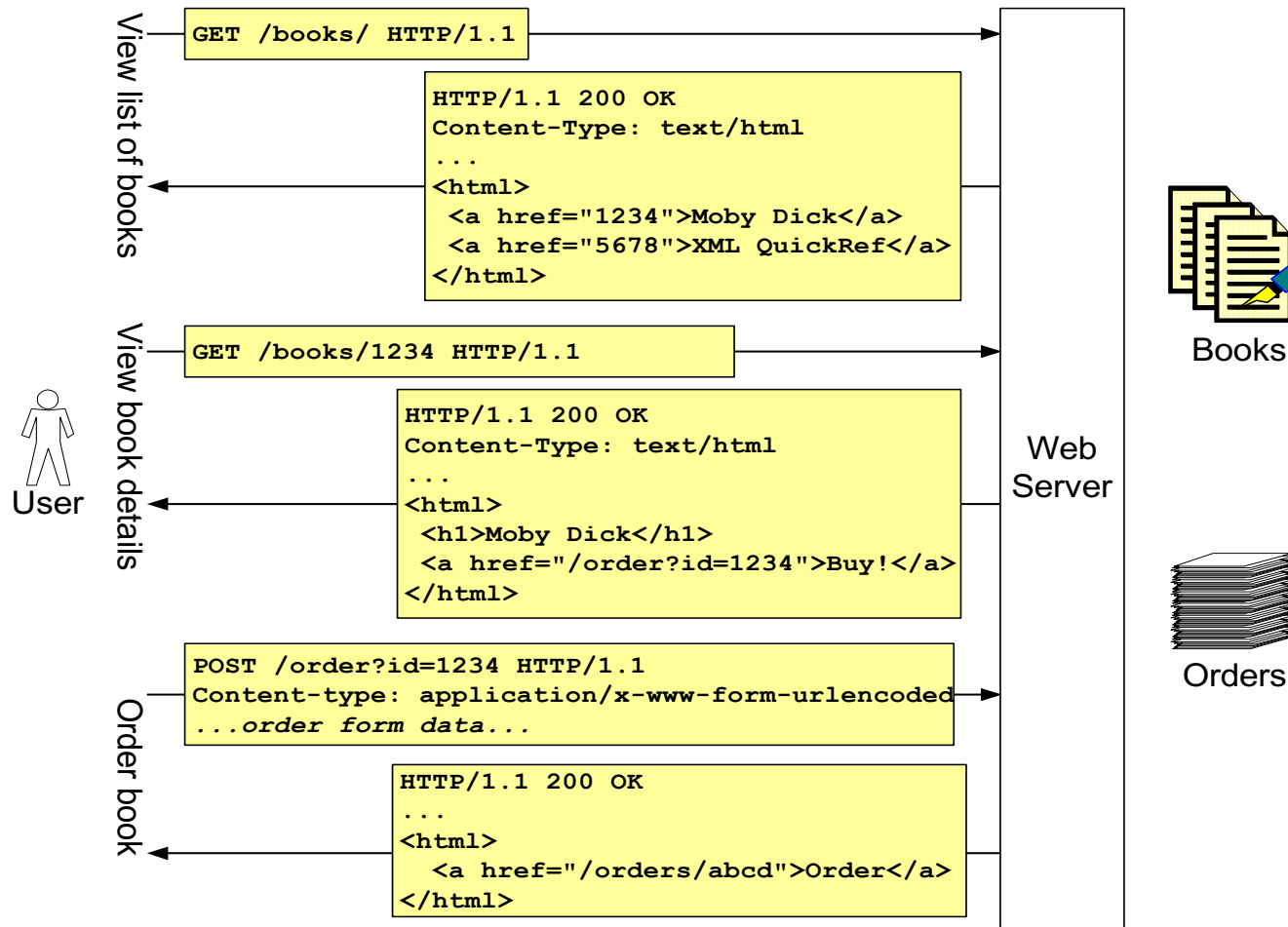
- RPC protocol encoded in JSON instead of XML
- Very simple protocol (and very similar to XML-RPC)
 - <https://www.jsonrpc.org/specification>

REST as Lightweight Web Services

- Much like Web Services, a REST service is:
 - **Platform-independent** (you don't care if the server is Unix, the client is a Mac, or anything else),
 - **Language-independent** (C# can talk to Java, etc.),
 - Standards-based (runs on top of **HTTP**), and
 - Can be used in the presence of **firewalls** (port 80/443 always open)
- Like Web Services, REST offers no built-in security features, encryption, session management, QoS guarantees, etc. But also as with Web Services, these can be added by building on top of HTTP:
 - For security, username/password tokens are often used.
 - For encryption, REST can be used on top of **HTTPS** (secure sockets).
- One thing that is *not* part of a good REST design is cookies:
 - The "ST" in "**REST**" stands for "State Transfer", and indeed, in a good REST design operations are self-contained, and each request carries with it (transfers) all the information (state) that the server needs in order to complete it.

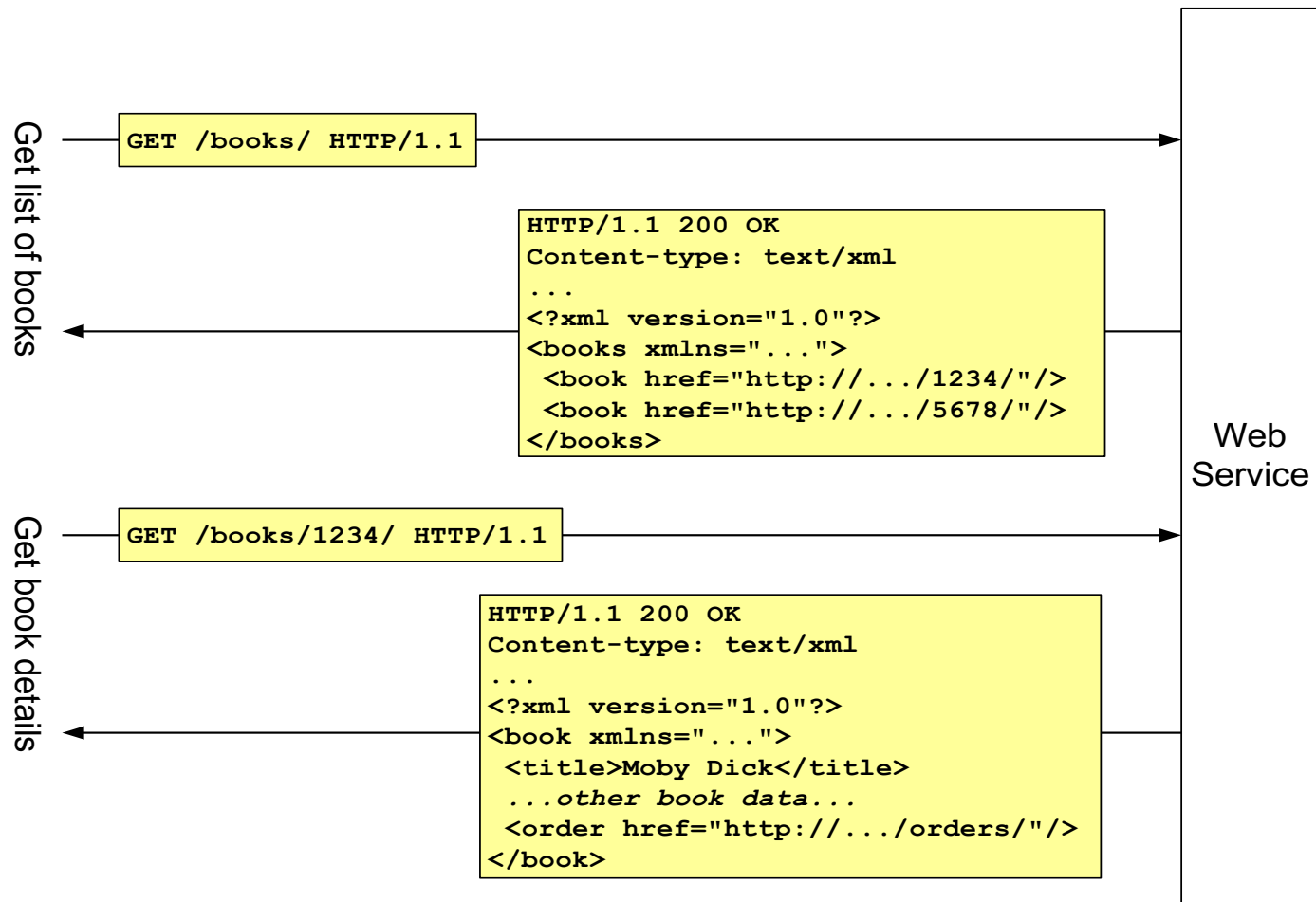
REST & the HTML Web

(Get book list, get book details, order book)



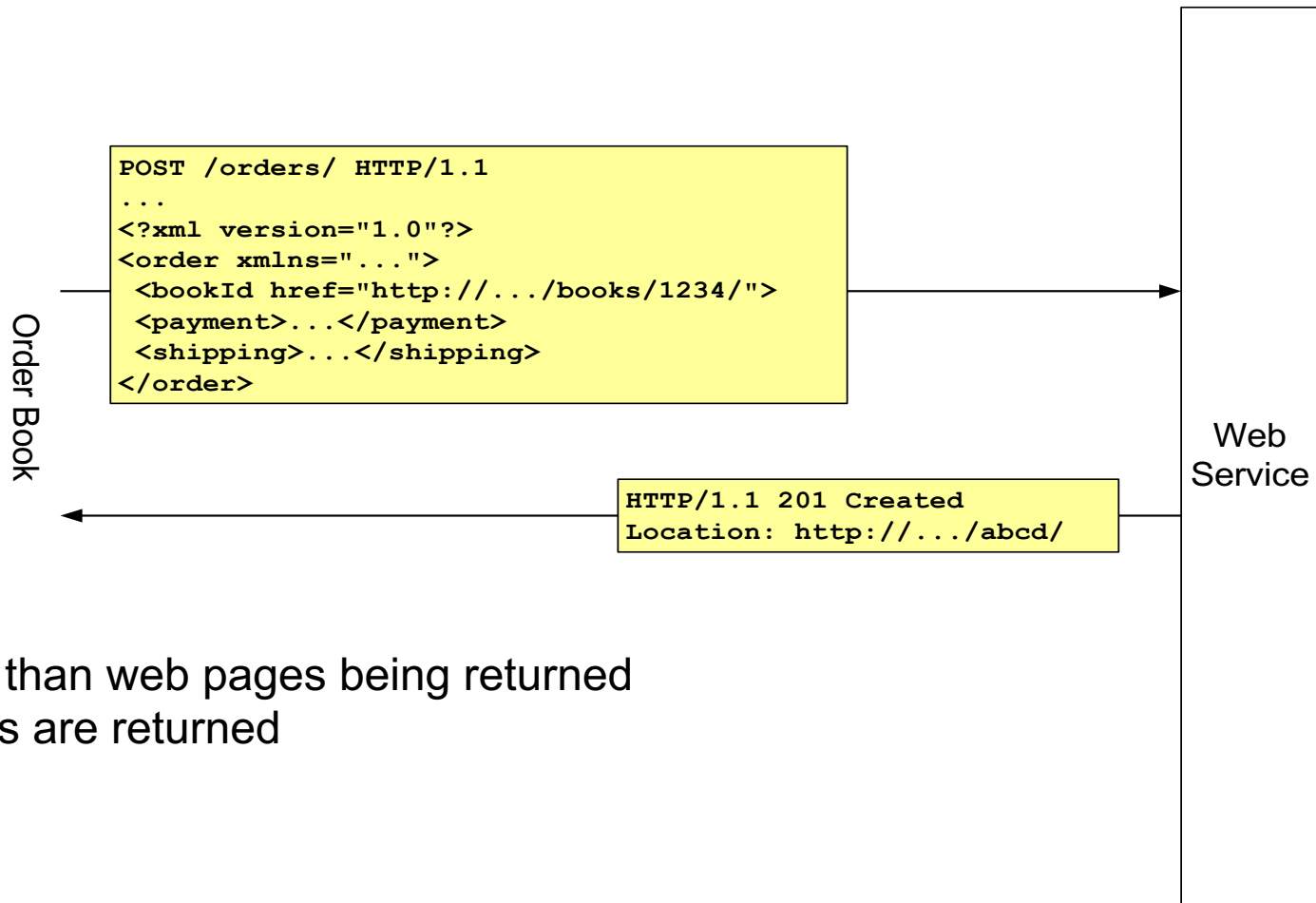
REST & the XML Web

(get book list, get book details)



REST & the XML Web (2)

(order book)



Rather than web pages being returned
xml files are returned

REST & the JSON Web

(get book list, get book details)

GET /books/ HTTP/1.1

HTTP/1.1 200 OK

Content-type: text/json

```
{  "books": {
    "book": [
      { "href": "http://.../1234/" },
      { "href": "http://.../5678/" }
    ]
  }
}
```

JSON objects are returned

GET /books/1234/ HTTP/1.1

HTTP/1.1 200 OK

Content-type: text/json

```
{
  "book": {
    "title": "Moby Dick",
    . . . other book data
    "order": { "href": "http://.../orders" }
  }
}
```

REST & the JSON Web (2)

(order book)

POST /orders/ HTTP/1.1

```
{  
  "order": {  
    "bookId": { "-href": "http://.../books/1234" },  
    "payment": " ... ",  
    "shipping": " ... "  
  }  
}
```

HTTP/1.1 201 Created

location: http://.../abcd

More Complex REST Requests

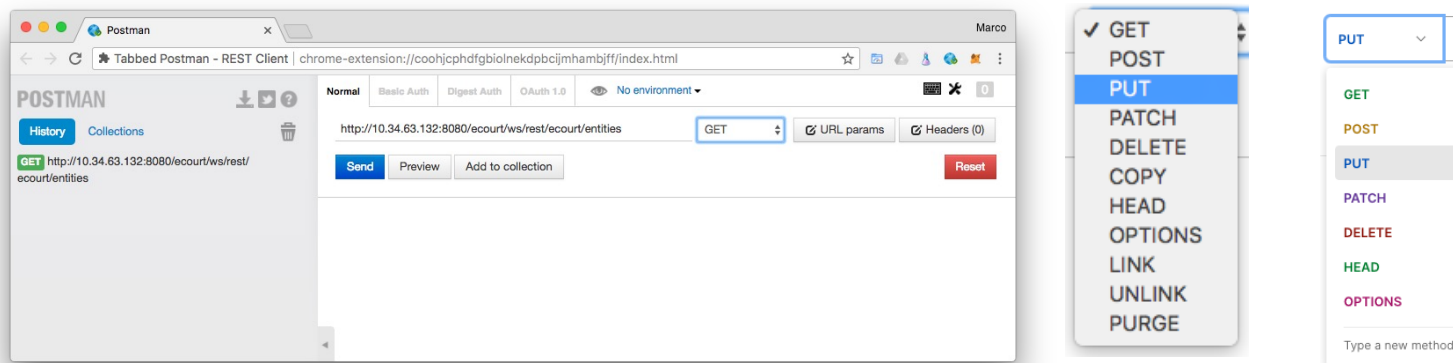
- REST can easily handle more complex requests, including multiple parameters.
- All types of HTTP requests: **GET, POST, PUT, PATCH, DELETE, HEAD, OPTIONS**
- Rarely used: **LINK, UNLINK, PURGE**
- In most cases, you'll just use HTTP GET parameters in the URL.
- For example:

`http://www.acme.com/phonebook/UserDetails?firstName=John&lastName=Doe`

- If you need to pass long parameters, or binary ones, you'd normally use HTTP POST requests, and include the parameters in the POST body.
- As a rule,
 1. **GET** requests should be for read-only queries; they should not change the state of the server and its data, List Items, for example
 2. For creation, updating, use **POST** requests. POST can also be used for read-only queries, as noted above, when complex params are required.'
 3. **PUT, DELETE** are also used for updating and deleting items.
- “Legacy” REST services might use XML in their responses.
- Newer REST Services use JSON in their responses.
- **Postman** can be used to test any of the HTTP requests.

Postman

- Postman is a tool for API testing
- Platforms include Chrome add-on, MacOS, Windows and Linux native apps
- Download page available at:
<https://www.getpostman.com/apps>
- Free version comes with the following support:
 - Unlimited Postman collections, variables, environments, & collection runs
 - Postman Workspaces
 - Postman Help Center & Community Support
 - API Documentation (1000 Monthly document views)
 - Mock Servers (1000 Monthly mock server calls)
 - Postman API (1000 Monthly API calls)
 - API Monitoring (100 Monthly calls)
- Postman Enterprise provide additional feature on a monthly subscription.



REST Server Responses

- A server response in REST used to be an XML file; for example,

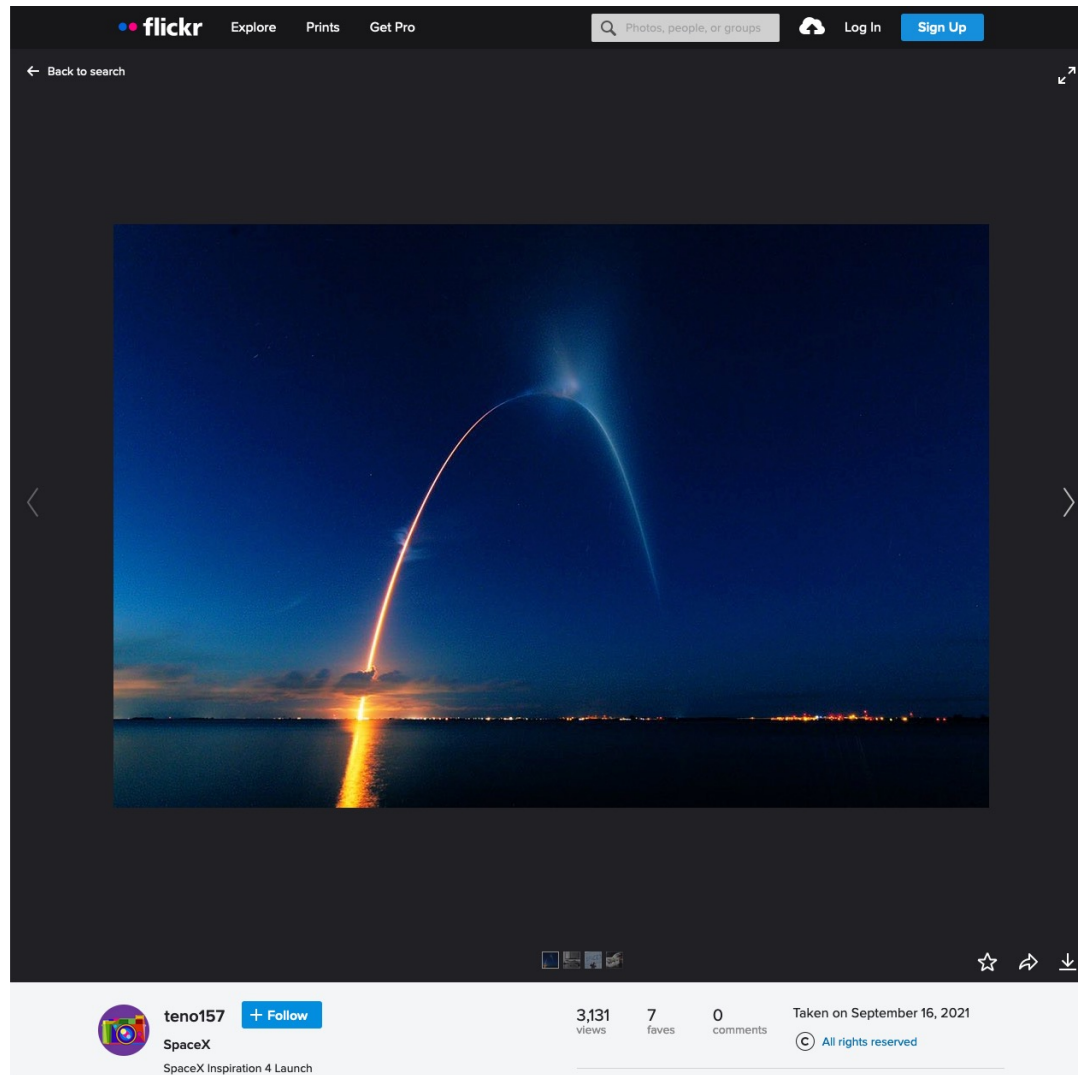
```
<parts-list>
  <part id="3322">
    <name>ACME Boomerang</name>
    <desc> Used by Coyote in <i>Zoom at the Top</i>, 1962 </desc>
    <price currency="usd" quantity="1">17.32</price>
    <uri>http://www.acme.com/parts/3322</uri> </part>
  <part id="783">
    <name>ACME Dehydrated Boulders</name>
    <desc> Used by Coyote in <i>Scrambled Aches</i>, 1957 </desc>
    <price currency="usd" quantity="pack">19.95</price>
    <uri>http://www.acme.com/parts/783</uri>
  </part> </parts-list>
```

- However, other formats can also be used; REST is *not* bound to XML in any way. JSON is the response format recently used the most. Possible formats include CSV.
- One option that is *not* acceptable as a REST response format, except in very specific cases is HTML, or any other format which is meant for human consumption and is not easily processed by clients.
- The specific exception is, of course, when the REST service is documented to return a human-readable document; and when viewing the entire WWW as a RESTful application, we find that HTML is in fact the most common REST response format.

Flickr

- Photo-sharing community with APIs provide viewing and uploading access
- See: <https://www.flickr.com/services/api/>
- **Request formats:** REST, XML-RCP, SOAP
- **Response Formats:** REST, XML-RPC, SOAP, JSON, PHP. Supports JSONP.
- API Developer Kits available for 15 languages including ActionScript (Flash), Java (Android), .NET, Objective-C (iOS)
- Comprehensive number of API methods for authentication, blogs, contacts, favorites, galleries, people, photos
- Example Query:
 - https://api.flickr.com/services/rest/?method=flickr.photos.getRecent&api_key=f2cc26448280a762143ba4a865795ab4&format=json
 - (remove format parameter for XML results)
 - **https** required since June 2014

Flickr



Flickr Sample JSON Result

```
jsonFlickrApi({"photos":{"page":1, "pages":10, "perpage":100, "total":1000, "photo":[{"id":"6879393174",
"owner":"50010354@N05", "secret":"cf784500dd", "server":"7080", "farm":8,
"title":"wjk_20110611_0092.jpg", "ispublic":1, "isfriend":0, "isfamily":0}, {"id":"6879393274",
"owner":"31403543@N03", "secret":"af280ab218", "server":"6231", "farm":7, "title":"Imagen 415",
"ispublic":1, "isfriend":0, "isfamily":0}, {"id":"6879393306", "owner":"66286618@N05",
"secret":"7fc731bc3d", "server":"6237", "farm":7, "title":"IMG_6241-1", "ispublic":1, "isfriend":0,
"isfamily":0}, {"id":"6879393338", "owner":"28935680@N03", "secret":"ec7444d9b6",
"server":"7237", "farm":8, "title":"IMG_6756", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"6879393352", "owner":"32752988@N06", "secret":"be56f5751c", "server":"6046", "farm":7,
"title":"AED_4586", "ispublic":1, "isfriend":0, "isfamily":0}, {"id":"6879393370",
"owner":"29083790@N00", "secret":"ec89570135", "server":"6219", "farm":7, "title":"IMG_6546",
"ispublic":1, "isfriend":0, "isfamily":0}, {"id":"6879393402", "owner":"50702313@N08",
"secret":"18ecdd7871", "server":"7191", "farm":8, "title":"Group A 3", "ispublic":1, "isfriend":0,
"isfamily":0}, {"id":"6879393418", "owner":"8502118@N08", "secret":"082968f6a9",
"server":"6220", "farm":7, "title":"Buff-necked Ibis (Theristicus caudatus)", "ispublic":1, "isfriend":0,
"isfamily":0}, {"id":"6879393440", "owner":"51425572@N04", "secret":"bc5f816ffb",
"server":"6219", "farm":7, "title":"P2115768", "ispublic":1, "isfriend":0, "isfamily":0}, [...]]})
```

Partial Flickr Sample JSON Result With Formatting

```
jsonFlickrApi({"photos":{"page":1, "pages":10, "perpage":100, "total":1000,
"photo":[
{"id":"6879682760", "owner":"8348059@N02", "secret":"1ac6c7e2c4", "server":"6220", "farm":7,
"title":"DSC_0619", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"6879682762", "owner":"35772789@N02", "secret":"db5dff91d", "server":"6117", "farm":7,
"title":"Dianna Romo 5", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"6879682776", "owner":"8091633@N05", "secret":"302174b53e", "server":"6118", "farm":7,
"title":"DSC_4259", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"6879682778", "owner":"58641881@N08", "secret":"c028082788", "server":"7212", "farm":8,
"title":"DSC_0777", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"6879682790", "owner":"32045507@N06", "secret":"d80d372bd2", "server":"6093", "farm":7,
"title":"IMG_9136", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"6879682792", "owner":"76919580@N08", "secret":"57e8d1cf8d", "server":"7277", "farm":8,
"title":"DSC01410", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"6879682796", "owner":"50838701@N04", "secret":"a3431e27e9", "server":"6042", "farm":7,
"title":"eP3274587", "ispublic":1, "isfriend":0, "isfamily":0},
```

Microsoft Bing Maps REST Services

- Bing Maps REST Services: <https://docs.microsoft.com/en-us/bingmaps/rest-services/>
- The Bing Spatial Data Services are REST-based services that offer three key functionalities: batch geocoding, point of interest (POI) data, and the ability to store and expose your spatial data.
- Used for performing tasks such as geocoding, reverse-geocoding, routing and static imagery.
- REST Request URLs:
 - Find a location by **Address**:
<http://dev.virtualsearth.net/REST/v1/Locations/CA/adminDistrict/postalCode/locality/addressLine?includeNeighborhood=includeNeighborhood&maxResults=maxResults&key=BingMapsKey>
 - Find a location by **Query**:
<http://dev.virtualsearth.net/REST/v1/Locations/1%20Microsoft%20Way%20Redmond%20WA%2098052?o=xml&key=BingMapsKey>
 - Find a location by **Point**:
<http://dev.virtualsearth.net/REST/v1/Elevation/List?points=35.89431,-110.72522,35.89393,-110.72578,35.89374,-110.72606,35.89337,-110.72662&key=BingMapsKey>

Microsoft REST Services (cont' d)

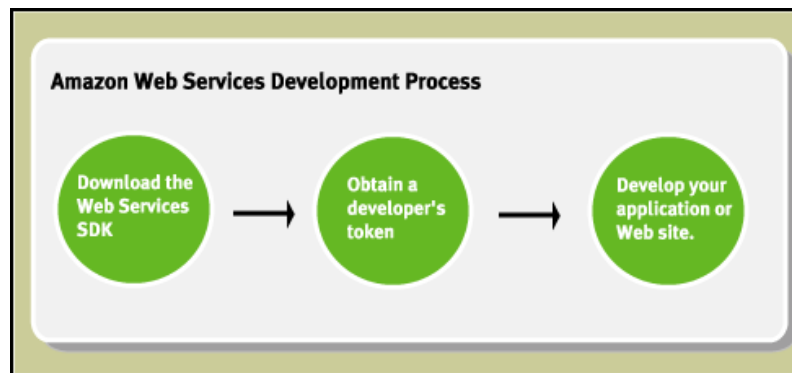
- Request Parameters: See complete list at:
<https://docs.microsoft.com/en-us/bingmaps/rest-services/common-parameters-and-types>
- **Response formats:** XML, JSON (output=JSON), JSONP (jsonp=callback), and PHP
- Response fields (Location Data) defined at:
<https://docs.microsoft.com/en-us/bingmaps/rest-services/common-response-description>

Amazon Web Services

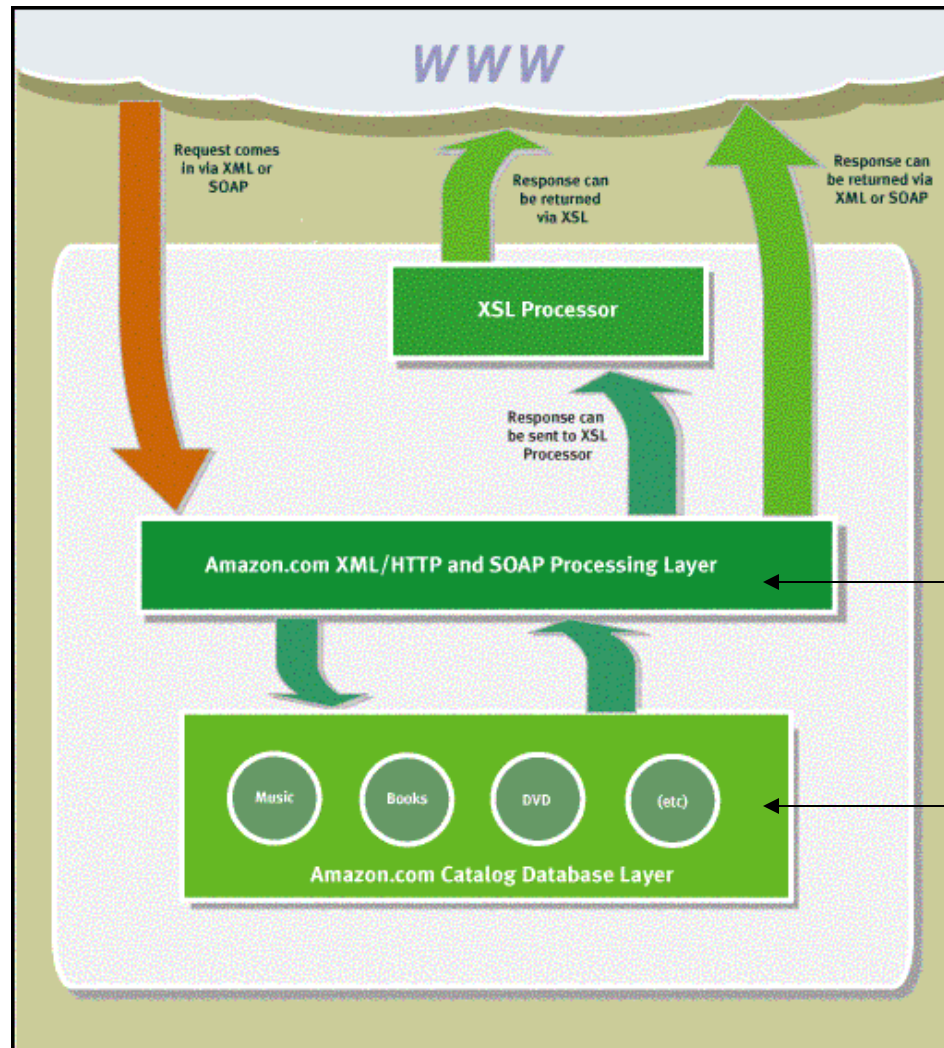
- Among them are
 - Amazon Associates Web Services (see next slides)
 - Amazon Elastic Compute Cloud (or EC2)
 - allows users to rent computers on which to run their own computer applications. EC2 allows scalable deployment of applications by providing a web service through which a user can boot an Amazon Machine Image to create a virtual machine, which Amazon calls an "instance", containing any software desired.
 - A user can create, launch, and terminate server instances as needed, paying by the hour for active servers, hence the term "elastic".
 - Amazon primarily charges customers in two ways:
 - On-demand pricing. Example: EC2 Linux, t2.nano \$0.0058 per hour
<https://aws.amazon.com/ec2/pricing/on-demand/>
 - Spot Instances pricing. Example: linux, m4.large \$0.019 per Hour
<https://aws.amazon.com/ec2/spot/pricing/>
 - Reserved Instances pricing.
<https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/>
 - Dedicated Hosts pricing.
<https://aws.amazon.com/ec2/dedicated-hosts/pricing/>
 - See http://en.wikipedia.org/wiki/Amazon_Elastic_Compute_Cloud for details

Amazon Associates Web Services

- Amazon offers web services to 3 types of users:
 - **Associates:** third-party site owners wishing to build more effective sponsored affiliate links to Amazon products, thus increasing their referral fees
 - **Vendors:** sellers on the Amazon platform looking to manage inventory and receive batch product data feeds
 - **Developers:** third-party developers building Amazon-driven functionality into their applications
- Amazon Web Services provides software developers direct access to Amazon's technology platform and product data.
- Developers can build businesses by creating Web sites and Web applications that use Amazon products, charging and delivery mechanisms.
- Using Web services, you can now enable your Web site visitors to add products to Amazon.com shopping carts, wedding registries, baby registries and wish lists directly from your site.
- <http://aws.amazon.com/>



Amazon Associates Web Services Data Flow



Amazon results can be returned either as XML files or as SOAP messages

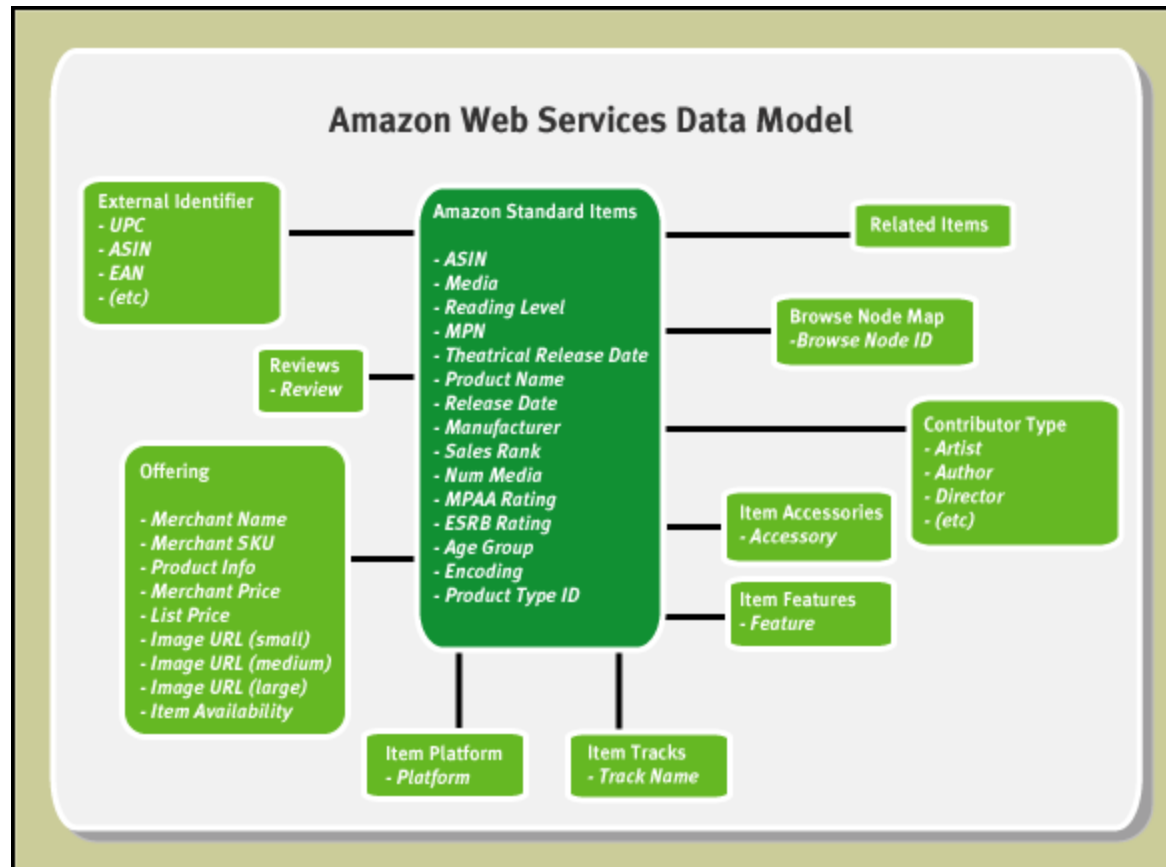
XML/HTTP stands for REST

Data

Amazon Web Services Data Model

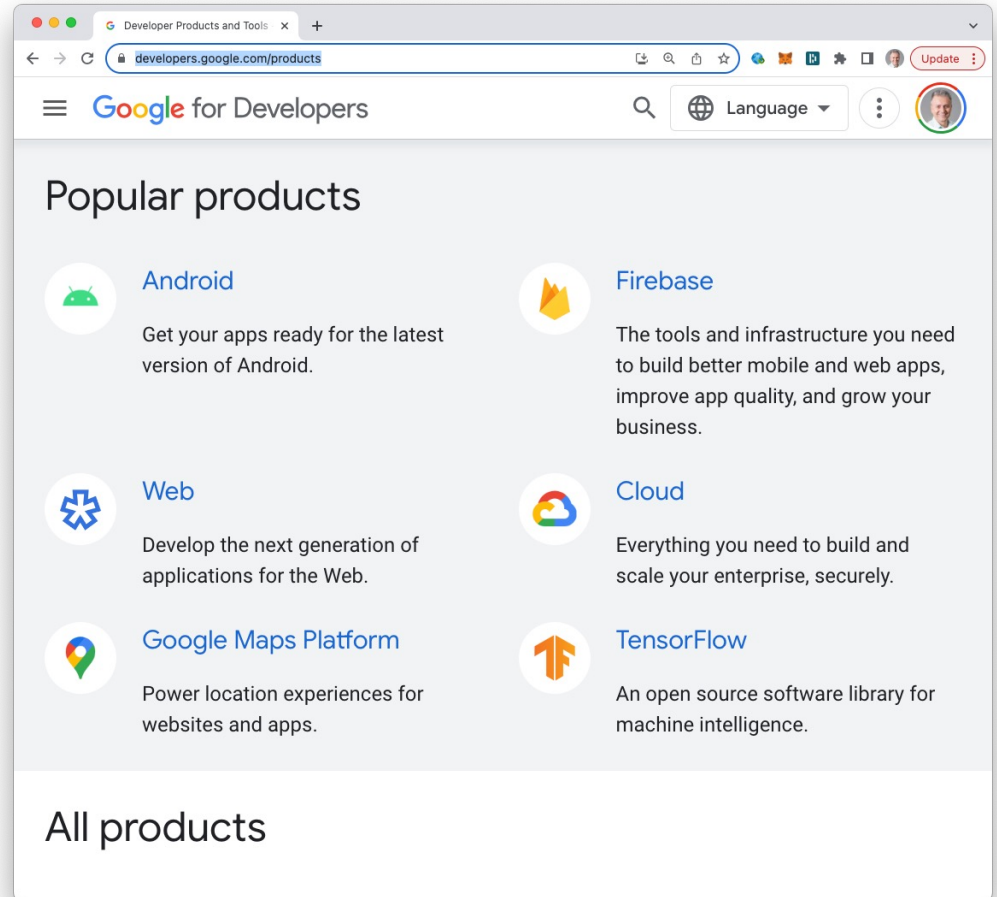
Below is a graphical listing of the elements of the Amazon Web Services data model.

The graphic represents the logical structure of AWS data.



Google APIs

- Available at:
 - <http://developers.google.com/products>
- APIs available for:
 - Android
 - App Engine
 - Chrome
 - Games
 - Google Maps
 - Google Apps
 - Google Play
 - Commerce
 - YouTube
 - Etc.
- Develop, Grow and Earn



Google App Engine

- Google App Engine lets you run Web Applications / Web Services on GCP.
 - There are no servers to maintain: You just upload your application (like AWS)
- You can serve your app from your own domain name, or you can serve your app using a free name on the appspot.com domain.
 - You can limit access to members of your organization.
- Google App Engine supports apps written in several programming languages
 1. **Java** environment, including the JVM, and Java servlets.
 2. **PHP**
 3. **Python**. App Engine also features two dedicated **Python** runtime environments, each of which includes a fast Python interpreter and the Python standard library.
 4. **Go**. App Engine provides a **Go** runtime that runs natively compiled Go code.
 5. **Node.js & Ruby**.
 6. **Custom Runtimes**. Included in the “flexible environment”.
- Download App Engine SDKs at:
<https://cloud.google.com/appengine/downloads>
- You only pay for what you use and there are no set-up costs and no recurring fees
- Free daily limits are quite high: **860K API calls (URLFetch API)** , 200h connect time, 5GB storage) See: <https://cloud.google.com/appengine/quotas>

Google App Engine Free Quota

- See: <https://cloud.google.com/appengine/quotas>

Resource	Free Default Limit	Billing Enabled Default Limit
Default Google Cloud Storage Bucket Stored Data	5 GB	First 5 GB free; no maximum
Default Google Cloud Storage Bucket Class A Operations	20,000 ops/day	First 20,000 ops/day free; no maximum
Default Google Cloud Storage Bucket Class B Operations	50,000 ops/day	First 50,000 ops/day free; no maximum
Default Google Cloud Storage Bucket Network Egress	Up to the Outgoing Bandwidth quota	Up to the Outgoing Bandwidth quota free; no maximum

Resource	Free Default Limit		Billing Enabled Default Limit	
	Daily Limit	Maximum Rate	Daily Limit	Maximum Rate
Channel API Calls	657,000 calls	3,000 calls/minute	91,995,495 calls	32,000 calls/minute
Channels Created	100 channels	6 creations/minute	Based on your spending limit	60 creations/minute
Channel Hours Requested	200 hours	12 hours requested/minute	Based on your spending limit	180 hours requested/minute
Channel Data Sent	Up to the Outgoing Bandwidth quota	22 MB/minute	1 TB	740 MB/minute

Resource	Cost
Code & Static Data Storage - First 1 GB	Free
Code & Static Data Storage - Exceeding 1 GB	\$0.026/GB/month

Resource	Free Default Daily Limit	Billing Enabled Default Limit
Stored Data (billable)	1 GB *	1 GB free; no maximum
Number of Indexes	200 *	200
Entity Reads	50,000	\$0.06/100k entity read
Entity Writes	20,000	\$0.18/100k entity writes
Entity Deletes	20,000	\$0.02/100k entity deletes
Small Operations	Unlimited	Not applicable

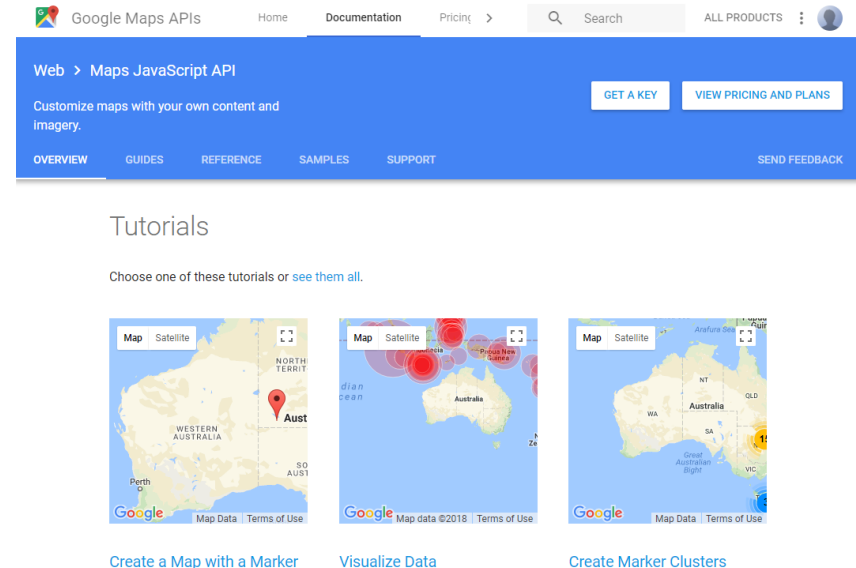
Programmable Search Engine

- Previously known as “Google Custom Search.”
- Enables searching over a website or a collection of websites
- Places a Google search box on a website that allows users to search the site
- Search results can be customized to match the design of the site
 - Google form to be filled out to create the custom search box
- <http://cse.google.com/>
- Create a “**Programmable search engine**”
 - Google Custom Search enables you to create a search engine for your website, your blog, or a collection of websites. You can configure your engine to search both web pages and images
- Search experience for users
 - Site search for your website
 - Topical search engine
 - Use structured data with Custom Search
- See: <https://developers.google.com/custom-search/>
- Documentation on implementing a “search box”:
<https://developers.google.com/custom-search/docs/tutorial/implementingsearchbox>

Google API Example: Google Maps API

- We will use the JavaScript API for Google maps
<https://developers.google.com/maps/documentation/javascript>
- We will use their API, V. 3, click on "Get Started"
- First step: obtain an API key by click on "GET A KEY"
- Second step: return to
<https://developers.google.com/maps/documentation/javascript/tutorial>
and examine the sample code

Note: Google Maps API material not required. Skip to Slide 43, Apple iCloud for developers.



Activate Google Maps JavaScript API v3

← Maps










🔍 Search for APIs & Services

Filter by

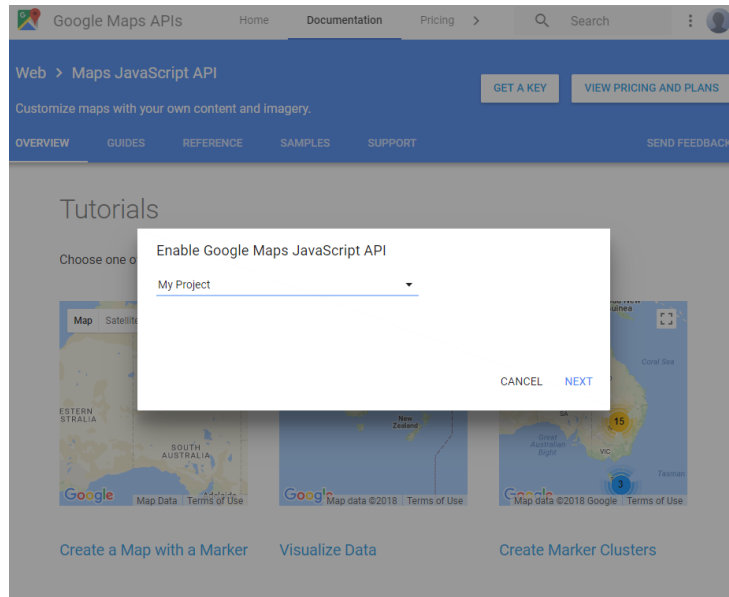
CATEGORY

Maps

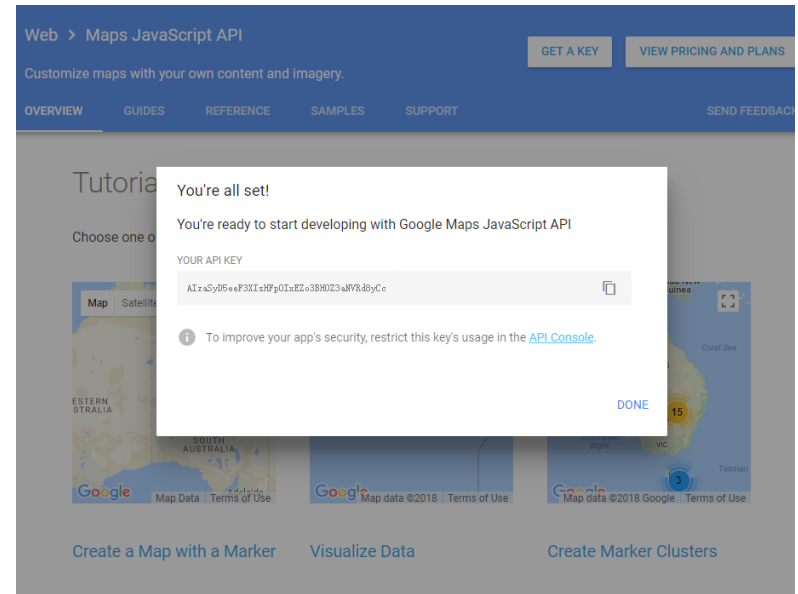
17 results

 Google Maps Android API Google Maps for your native Android app.	 Google Maps Directions API Google Directions between multiple locations.	 Google Maps Distance Matrix API Google Travel time and distance for multiple destinations.
 Google Maps Elevation API Google Elevation data for any point in the world.	 Google Maps Embed API Google Make places easily discoverable with interactive Google Maps.	 Google Maps Geocoding API Google Convert between addresses and geographic coordinates.
 Google Maps Geolocation API Google Location data from cell towers and WiFi nodes.	 Google Maps JavaScript API Google Maps for your website	 Google Maps Roads API Google Snap-to-road functionality to accurately trace GPS breadcrumbs.

Obtaining an API Key



Select Project



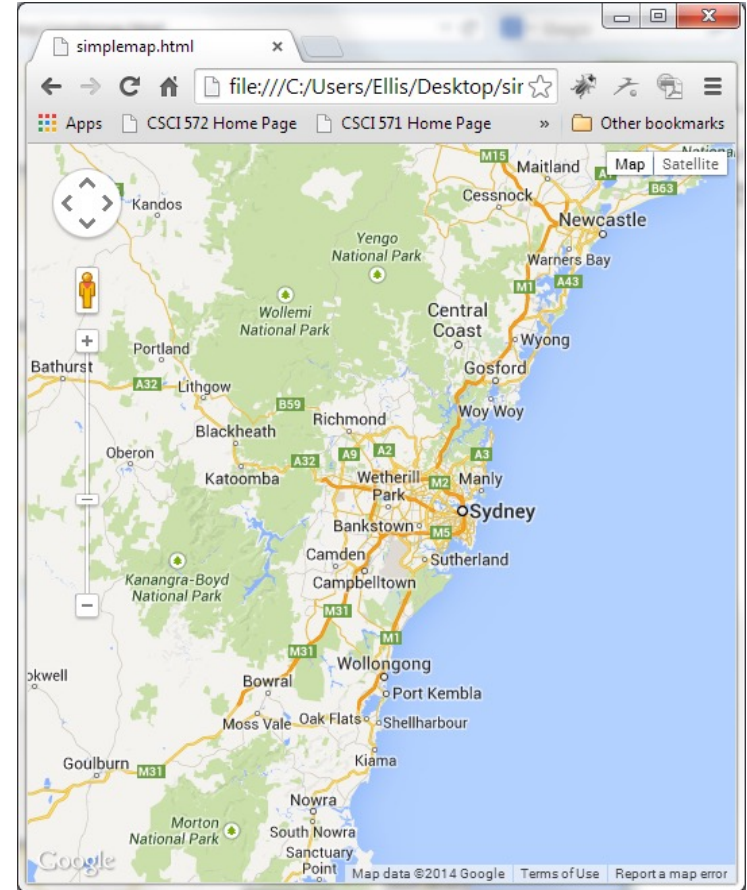
Returning a key result

Simple Maps Example

```
<!DOCTYPE html>

<html><head><meta name="viewport" content="initial-
scale=1.0, user-scalable=no" />

  <style type="text/css">
    html { height: 100% }
    body { height: 100%; margin: 0; padding: 0 }
    #map-canvas { height: 100% }
  </style>
  <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=API_KEY"
>
  </script>
  <script type="text/javascript">
    function initialize() {
      var mapOptions = {
        center: new google.maps.LatLng(-34.397,
150.644),
        zoom: 8    };
      var map = new
google.maps.Map(document.getElementById("map-canvas"),
        mapOptions);    }
      google.maps.event.addDomListener(window, 'load',
initialize);</script></head>
<body> <div id="map-canvas"/></body></html>
```



for many more details about this example see
<https://developers.google.com/maps/documentation/javascript/tutorial>

Changing the Map's Center Point

- Use Geocoding API to find the latitude/longitude of a local address
- Use a geocoding service at:
<https://developers.google.com/maps/documentation/geocoding/start>
- For an address we will use the CS dept
- The result is the lat/long

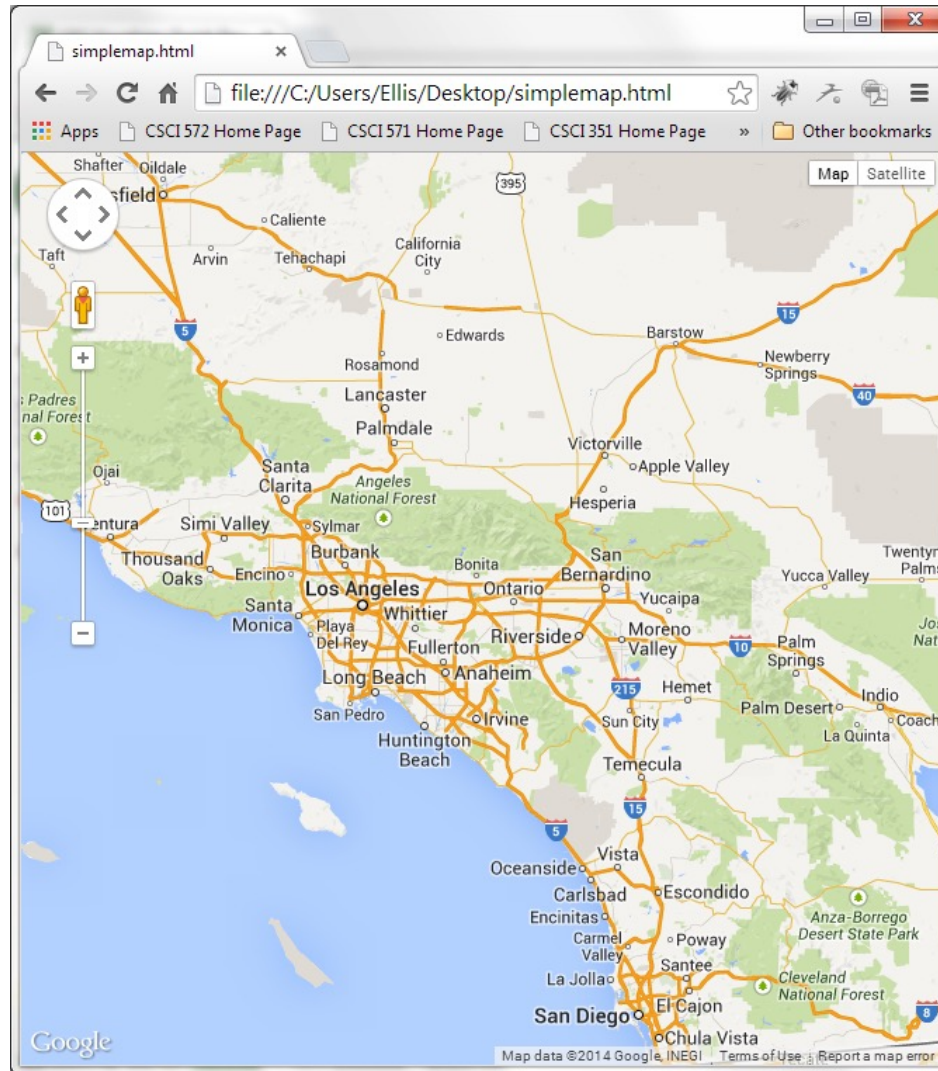
The image shows a screenshot of the Google Maps Geocoding API documentation page. The top navigation bar includes 'Web Services > Geocoding API', 'GET A KEY', 'VIEW PRICING AND PLANS', 'GUIDES', 'SUPPORT', and 'SEND FEEDBACK'. The left sidebar lists 'Get Started' (Developer's Guide, Geocoding Best Practices, Geocoder FAQ, Best Practices for Web Services, Client Libraries, Get API Key), 'Policies and Terms', 'Usage Limits', 'Optimizing Quota Usage', and 'Policies'. The main content area is titled 'Getting Started' with a star rating and a 'Contents' dropdown. The contents list includes 'Sample request and response', 'Geocoding request and response (latitude/longitude lookup)', 'Reverse geocoding request and response (address lookup)', and 'Start coding with our client libraries'. Below this, a paragraph states: 'The Google Maps Geocoding API is a service that provides geocoding and reverse geocoding of addresses.'

Below the documentation, a browser window shows a REST client interface. The URL bar displays `https://maps.googleapis.com/maps/api/geocode/json?address=941+bloom+walk+Los+Angeles,+CA&`. The 'JSON' tab is selected, showing a response with the following structure:

```
{
  "results": [
    {
      "address_components": [...],
      "formatted_address": "Salvatori Computer Science Center, 941 Bloom Walk, Los Angeles, CA 90089, USA",
      "geometry": {
        "bounds": {...},
        "location": {
          "lat": 34.019476,
          "lng": -118.2894739,
          "location_type": "ROOFTOP",
          "viewport": {...}
        },
        "place_id": "ChIJ4Tfr1PzHwoARkKpnxY2btw4"
      },
      "types": [...]
    }
  ],
  "status": "OK"
}
```

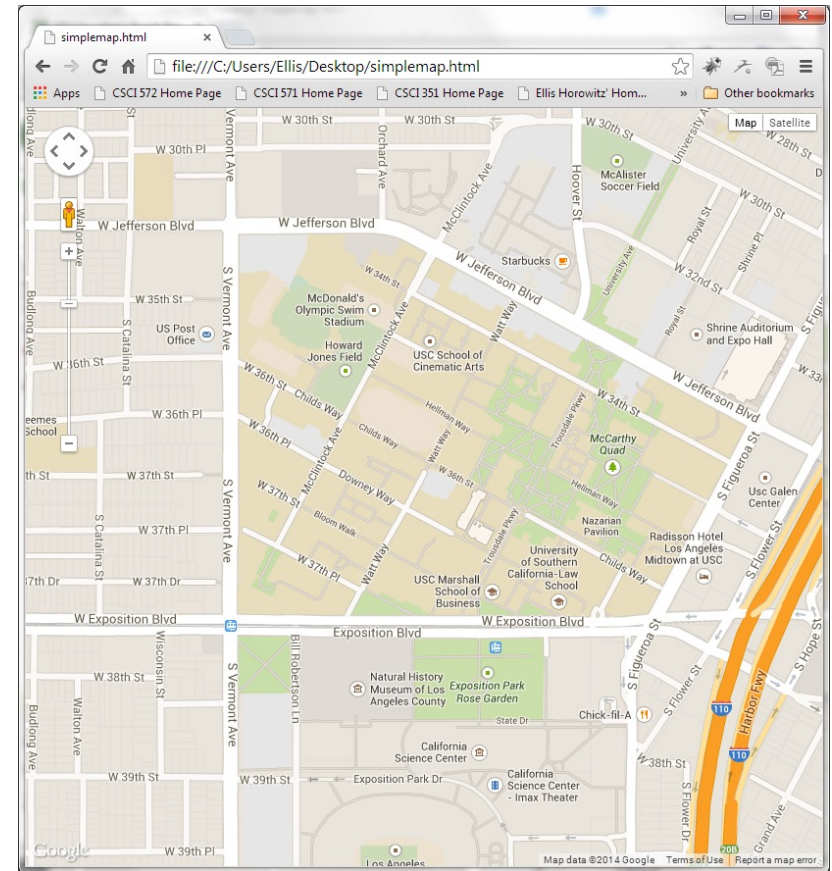
Arrows from the text 'For an address we will use the CS dept' and 'The result is the lat/long' point to the 'formatted_address' and 'location' fields in the JSON response, respectively.

Simple Map with Lat/Long Change



Changing the Zoom Level

- the zoom level controls the distance above the map
- higher values cause the zoom to close in
- set the zoom value to 16 and the resulting map is produced



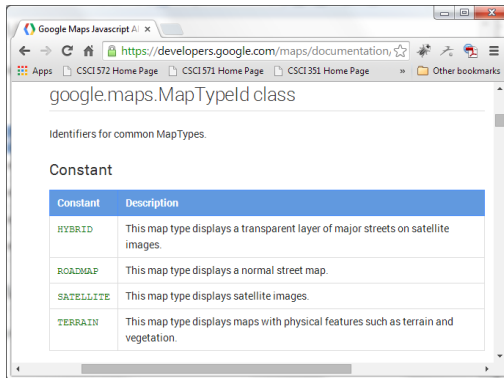
ROADMAP

Adding a Marker to the Map

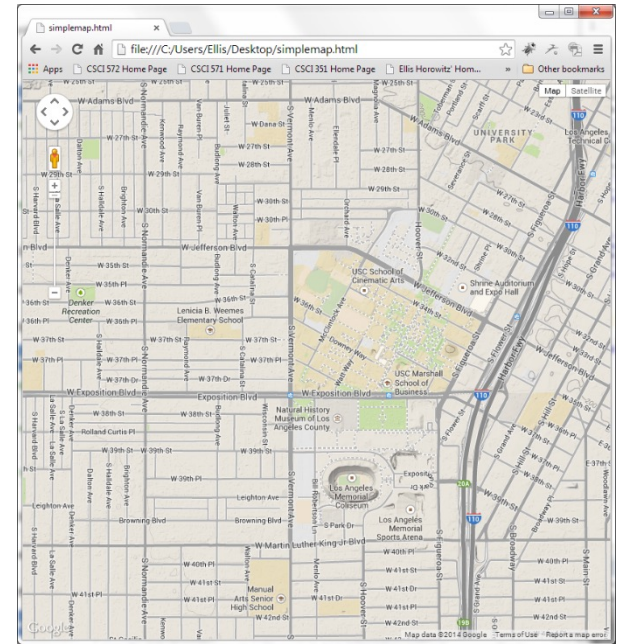
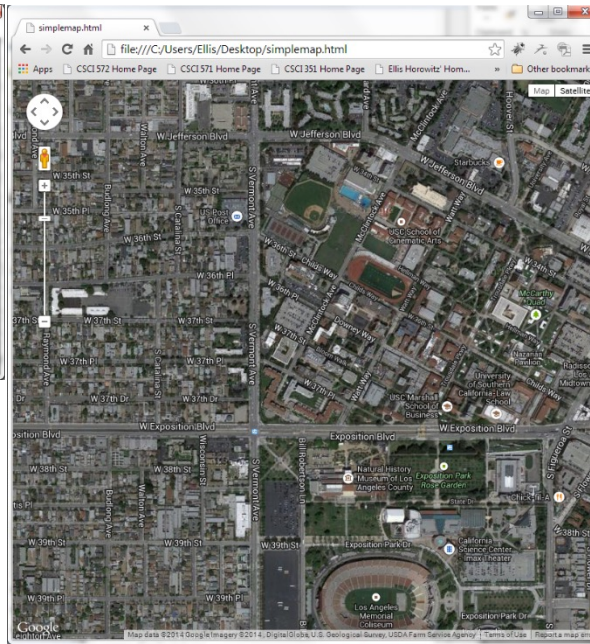
- But where is the CS dept.? we need to add a marker
- we see an example of a marker at <https://developers.google.com/maps/documentation/javascript/examples/marker-simple>

```
function initialize() {  
    var myLatLng = {lat: 34.020, lng: -118.290};  
    var mapOptions = { zoom: 4, center: myLatLng }  
    var map = new google.maps.Map(document.getElementById('map-  
canvas'), mapOptions);  
    var marker = new google.maps.Marker({  
        position: myLatLng,  
        map: map,  
        title: 'CS Dept'  
    });  
}  
google.maps.event.addDomListener(window, 'load', initialize);
```


Change the Map Type



there are 4 map types:
HYBRID
ROADMAP
SATELLITE
TERRAIN



one can alter the map type by adding the line:
`mapTypeId: 'satellite';` or
`map.setMapTypeId('terrain');`

Add a Marker with Tool Tip

```
<!DOCTYPE html>

<html><head><meta name="viewport" content="initial-scale=1.0,
user-scalable=no" />

<style type="text/css">

    html { height: 100% } body { height: 100%; margin: 0;
padding: 0 }

    #map-canvas { height: 100% }

</style> <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyC0CF4Fh0OQs
redZjT8LA16yvZ7Ux9dnuQ">

</script>

<script type="text/javascript">

function initialize() {

var mapOptions = {center: new google.maps.LatLng(34.020, -
118.290),

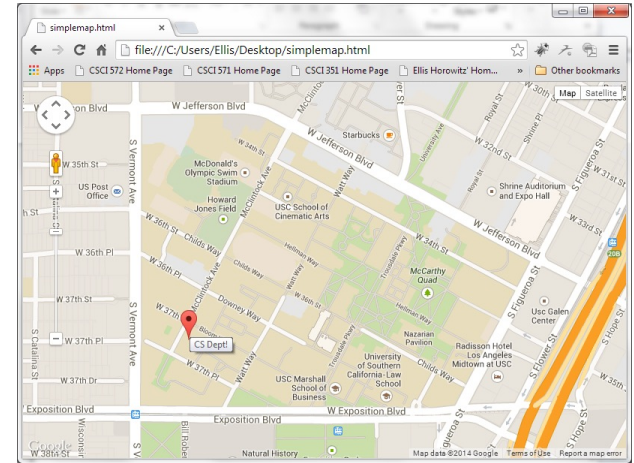
    zoom: 16    };

var map = new google.maps.Map(document.getElementById("map-
canvas"), mapOptions);

    var marker = new google.maps.Marker({
        position: new google.maps.LatLng(34.020, -118.290),
        map: map,
        title: 'CS Dept!'    })    }

    google.maps.event.addDomListener(window, 'load',
initialize);</script></head>

<body> <div id="map-canvas"/></body></html>
```

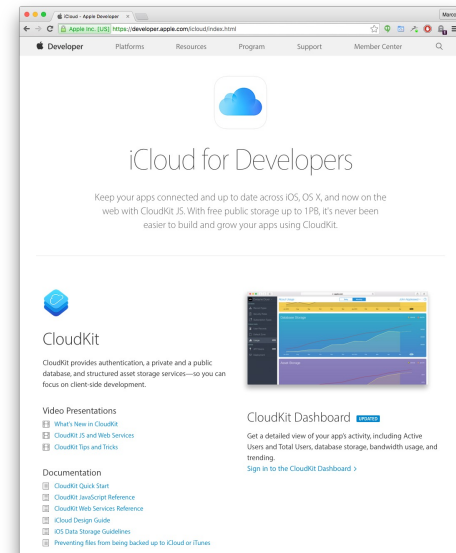


Adding a Popup Info Window to the Marker

```
<!DOCTYPE html><html><head><meta name="viewport" content="initial-
scale=1.0, user-scalable=no" />
    <style type="text/css">
html { height: 100% } body { height: 100%; margin: 0; padding: 0 }
#map-canvas { height: 100% }
    </style><script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?key=AIzaSyC0CF4Fh0OQsredZjT8LA
16yvZ7Ux9dnuQ">
    </script><script type="text/javascript">
function initialize() {
var mapOptions = {
center: new google.maps.LatLng(34.020, -118.290),zoom: 16    };
var map = new google.maps.Map(document.getElementById("map-canvas"),
mapOptions);
var marker = new google.maps.Marker({
    'position': new google.maps.LatLng(34.020, -118.290),
'map': map, 'title': 'CS Dept!'))}
var contentString = '<div id="content">'+
'<div id="siteNotice">CS Dept</div></div>';
var infowindow = new google.maps.InfoWindow({ content: contentString });
    google.maps.event.addDomListener(window, 'load', initialize);
    google.maps.event.addListener(marker, 'click', function() {
infowindow.open(map, marker) } );
</script></head><body> <div id="map-canvas"/></body></html>
```

Apple iCloud For Developers

- Apple's iCloud service places all information captured on any Apple device into the cloud, making it immediately available to all other Apple devices
- 5GB (free) – 50GB, 200GB, 1TB plans available at:
 - <http://www.apple.com/icloud/>
 - <https://developer.apple.com/icloud/index.html>
- iCloud APIs available for iOS 5 and OS X 10.9+
 - CloudKit framework
 - Storage API for Documents
 - Storage API for key-value data storage
 - Storage API for Core Data
 - Fallback Store (iOS 7+)
 - Account Changes (iOS 7+)
 - Manage iCloud Content (iOS 7+)
 - Xcode debugging (Xcode 5+)
 - iPhone simulator support (iOS 7+)



REST Best Practices

- 1. Provide a **URI for each resource** that you want exposed.
- 2. Prefer URIs that are logical over URIs that are physical. For example, prefer <http://www.boeing.com/airplanes/747>

Over:

<http://www.boeing.com/airplanes/747.html>

- Logical URIs allow the resource implementation to change without impacting client applications
- 3. As a corollary to (2) **use nouns in the logical URI, not verbs**. Resources are "things" not "actions"
- 4. Make all HTTP GETs side-effect free.
- 5. Use links in your responses to requests. Doing so connects your response with other data. It enables client applications to be self-propelled. That is, the response itself contains info about "what's the next step to take".
- 6. **Minimize the use of query strings**. For example, prefer

<http://www.parts-depot.com/parts/00345>

Over

<http://www.parts-depot.com/parts?part-id=00345>

- 7. Use the slash "/" to represent a parent-child, whole-part relationship
- 8. Use a "gradual unfolding methodology" for exposing data to clients. That is, a resource representation should provide links to obtain more details.
- 9. Always implement a service **using HTTP GET** when the purpose of the service is to allow a client to **retrieve a resource representation**, i.e., don't use HTTP POST