# APIs & Web Services

For Web Developers

# API

API stands for **A**pplication **P**rogram **I**nterface.

Think of it as an app-to-app interaction that is typically used to:

- Send data
- Receive data
- Update data

# Types of APIs

There are several ways that two applications can communicate:

- Library based APIs.  (like Google Maps)
- Class based APIs.  (like Android API )
- OS functions and routines.
- Hardware APIs (drivers)
- Web Services ( over the Web )

# Types of Web Services

An API for a website is code that allows two software programs to communicate with each other, over the Web.

- WSDL
- RSS
- SOAP
- REST

# Communication over the Web

Programs exchange data in the form of:

- Plain Text
- CSV
- JSON
- XML
- RSS
- and more..

# JSON

**JS Object ( Parsed )**

{ name: ‘Alex’, age: 30 }

**JSON ( Stringified )**

{ “name”: “Alex”, “age”, “30” }

**JSON Conversion:**

```
var obj = JSON.parse( str );
var str =  JSON.stringify( obj);
```

# XML

XML stands for eXtensible Markup Language.

- XML was designed to store and transport data.
- XML was designed to be both human- and machine-readable.
- An XML document is a string of characters. Almost every legal Unicode character may appear in an XML document.
- A tag is a markup construct that begins with **<** and ends with **>**

# XML Example

http://www.boi.org.il/currency.xml

```xml
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<CURRENCIES>
  <LAST_UPDATE>2017-06-06</LAST_UPDATE>
  <CURRENCY>
    <NAME>Dollar</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>USD</CURRENCYCODE>
    <COUNTRY>USA</COUNTRY>
    <RATE>3.545</RATE>
    <CHANGE>-0.085</CHANGE>
  </CURRENCY>
  <CURRENCY>
    <NAME>Pound</NAME>
    <UNIT>1</UNIT>
    <CURRENCYCODE>GBP</CURRENCYCODE>
    <COUNTRY>Great Britain</COUNTRY>
    <RATE>4.5734</RATE>
    <CHANGE>-0.114</CHANGE>
  </CURRENCY>
```

# XML and PHP

**Parse XML in PHP**

```
// Convert xml string into an SimpleXML Object
$xmlObj = simplexml_load_string( $st );

echo $xmlObj->all->person->name;

// tip
var_dump( $xmlObj );
```

# xml2js (NodeJS)

No extensive tutorials required because you are a smart developer! The task of parsing XML should be an easy one, so let's make it so! ( leonidas, npm )

```
var parseString = require('xml2js').parseString;

var xml = "<root>Hello xml2js!</root>";
parseString(xml, function (err, result) {
    console.log(result);
});
```

# RSS

RSS stands for Really Simple Syndication.

- RSS allows you to syndicate your site content
- RSS defines an easy way to share and view headlines and content.
- RSS files can be automatically updated.
- RSS is written in XML.

# RSS Structure

The first line in the document - the XML declaration - defines the XML version and the character encoding used in the document.

The next line is the RSS declaration which identifies that this is an RSS document (in this case, RSS version 2.0).

The next line contains the <channel> element.

# RSS Structure

The <channel> element has three required child elements:

<title> - Defines the title of the channel
<link> - Defines the hyperlink to the channel
<description> - Describes the channel

Each <channel> element can have one or more <item> elements.

# RSS Structure

The <item> element has three required child elements:

<title> - Defines the title of the item
<link> - Defines the hyperlink to the item
<description> - Describes the item

Finally, the two last lines close the <channel> and <rss> elements.

# RSS Structure

Because RSS is XML, keep in mind that:

- All elements must have a closing tag.
- Elements are case sensitive.
- Elements must be properly nested.
- Attribute values must always be quoted.

# RSS Example

http://www.ynet.co.il/Integration/StoryRss546.xml

# SOAP

SOAP stands for **S**imple **O**bject **A**ccess **P**rotocol

- SOAP is based on XML
- SOAP is a W3C recommendation
- SOAP brings it’s own protocol and focuses on exposing pieces of application logic (not data) as services.
- SOAP is not only for Web.

# REST

It stands for **RE**presentational **S**tate **T**ransfer.

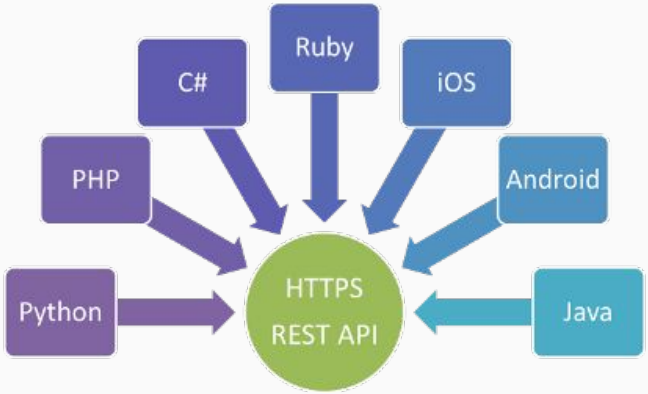
A RESTful API is an application program interface (API) that uses HTTP requests to GET, PUT, POST and DELETE data.

# REST

REST uses standard HTTP to handle CRUD operations on data it is much simpler in just about every other way.

Creating clients, developing APIs, the documentation is much easier to understand and REST can do almost anything easier and better than SOAP.

# REST




REST permits many different data formats whereas SOAP only permits XML.

# Data as Text

When exchanging data between a browser and a server, the data can only be text.

CSV, XML, RSS, JSON are all text that need to be converted into objects.

The great advantage of JSON is that we can work with the data with no complicated parsing and translations.

# Server as a Client

In Web APIs, sometimes a Server needs to behave “as a client” to another server.

# Server as a Client in NodeJS

In NodeJS there are several modules that handle http calls.

```
var request = require('request');

request('http://www.google.com', function (error, response, body) {
  console.log('error:', error); // Print the error if one occurred
  console.log('statusCode:', response && response.statusCode); // Print the response status code if a response was received
  console.log('body:', body); // Print the HTML for the Google homepage.
});
```

Read more: https://www.npmjs.com/package/request

# Server as a Client in PHP

In PHP you can use `file_get_contents( url )` or `cURL`.

**Example**:

```
$html = file_get_contents('http://www.google.com/');
```

# Using cURL in PHP

## cURL Example:

```
//  Data to send
$postData = "name=Alex&age=30";

//  Actually sending the data
$ch = curl_init( $url );
curl_setopt( $ch, CURLOPT_URL, $url );
curl_setopt( $ch, CURLOPT_POST, true);
curl_setopt( $ch, CURLOPT_POSTFIELDS, $postData );
curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true); //  Return response
curl_setopt( $ch, CURLOPT_SSL_VERIFYPEER, false);
curl_setopt( $ch, CURLOPT_FAILONERROR, false);
curl_setopt( $ch, CURLOPT_HTTPHEADER, array("charset=utf-8") );

//  Send request and get response
$response = curl_exec( $ch );
```

# Using cURL in PHP

As easy as this:

```
var curl = require('node-curl');

curl('www.google.com', function(err) {
  console.info(this.status);
  console.info(this.body);

});
```

Read More: https://www.npmjs.com/package/node-curl

# Programmable Web

APIs for Web Developers

# Useful APIs

XML stands for eXtensible Markup Language.

- Weather API - https://openweathermap.org/api
- IP to Country
  https://www.maxmind.com/en/geoip2-precision-country-service
- and More: **https://www.programmableweb.com/**

# Social Networks APIs

XML stands for eXtensible Markup Language.

- **Instagram** https://www.instagram.com/developer/
- **Twitter** https://dev.twitter.com/rest/public
- **Youtube** https://developers.google.com/youtube/iframe_api_reference
- **Facebook Login** https://developers.facebook.com/docs/facebook-login