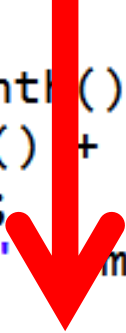


CS371m - Mobile Computing

WebView and Web Services

Using Built In Browser App

- To use the built in browser create an Intent and start the Activity



```
public void showTop10(View v) {  
    int day = datePicker.getDayOfMonth();  
    int month = datePicker.getMonth() + 1;  
    int year = datePicker.getYear();  
    Log.d(TAG, "date: " + day + "\\\" month + "\\\" + year);  
  
    Intent i = new Intent(Intent.ACTION_VIEW,  
        Uri.parse("http://www.cbs.com/late_night" +  
            "/late_show/top_ten/" +  
            "top_ten_update_by_date.php?year="+year  
            +"&month=" + month + "&day=" + day));  
    startActivity(i);  
}
```

WebView

- A View that display web pages
 - basis for creating your own web browser
 - OR just display some online content inside of your Activity
- Uses WebKit rendering engine
 - <http://www.webkit.org/>



The WebKit Open Source Project

WebView

- Android 4.4, API level 19 added an alternative to WebKit
- Chromium
- "Chromium WebView provides broad support for HTML5, CSS3, and JavaScript. It supports most of the HTML5 features available in Chrome for Android 30. It also brings an updated version of the JavaScript Engine (V8) that delivers dramatically improved JavaScript performance."



WebView

- Built in functionality to:
- display page
- navigate forward and backwards through a history
- zoom in and out
- perform searches
- and more:
 - capture images of page, search page for string, deal with cookies on a per application basis,

More on WebView

- Scenarios for using WebView in app instead of built in browser:
- provide info the app might need to update such as end user agreement or user guide (instead of doing app update)
 - display documents hosted online
- OR application provides data that ALWAYS requires internet connect to retrieve data
 - as opposed to performing network request and parsing data to display in Android layout
- OR display ads (blah)
- <http://developer.android.com/guide/webapps/webview.html>

WebView Example

- Simple app to view and navigate web pages - demo WebView class
- res/layout/main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<WebView xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/webview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
/>
```

WebView Activity

- override onCreate
- go to UT mobile site

```
public class HelloWebView extends Activity {  
  
    private WebView mWebView;  
  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.main);  
  
        mWebView = (WebView) findViewById(R.id.webview);  
        mWebView.getSettings().setJavaScriptEnabled(true);  
        mWebView.loadUrl("http://m.utexas.edu");  
    }  
}
```


WebView Example

- Must add permission for app to use Internet
- Also change style so no title bar

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="scottm.examples"
    android:versionCode="1"
    android:versionName="1.0" >

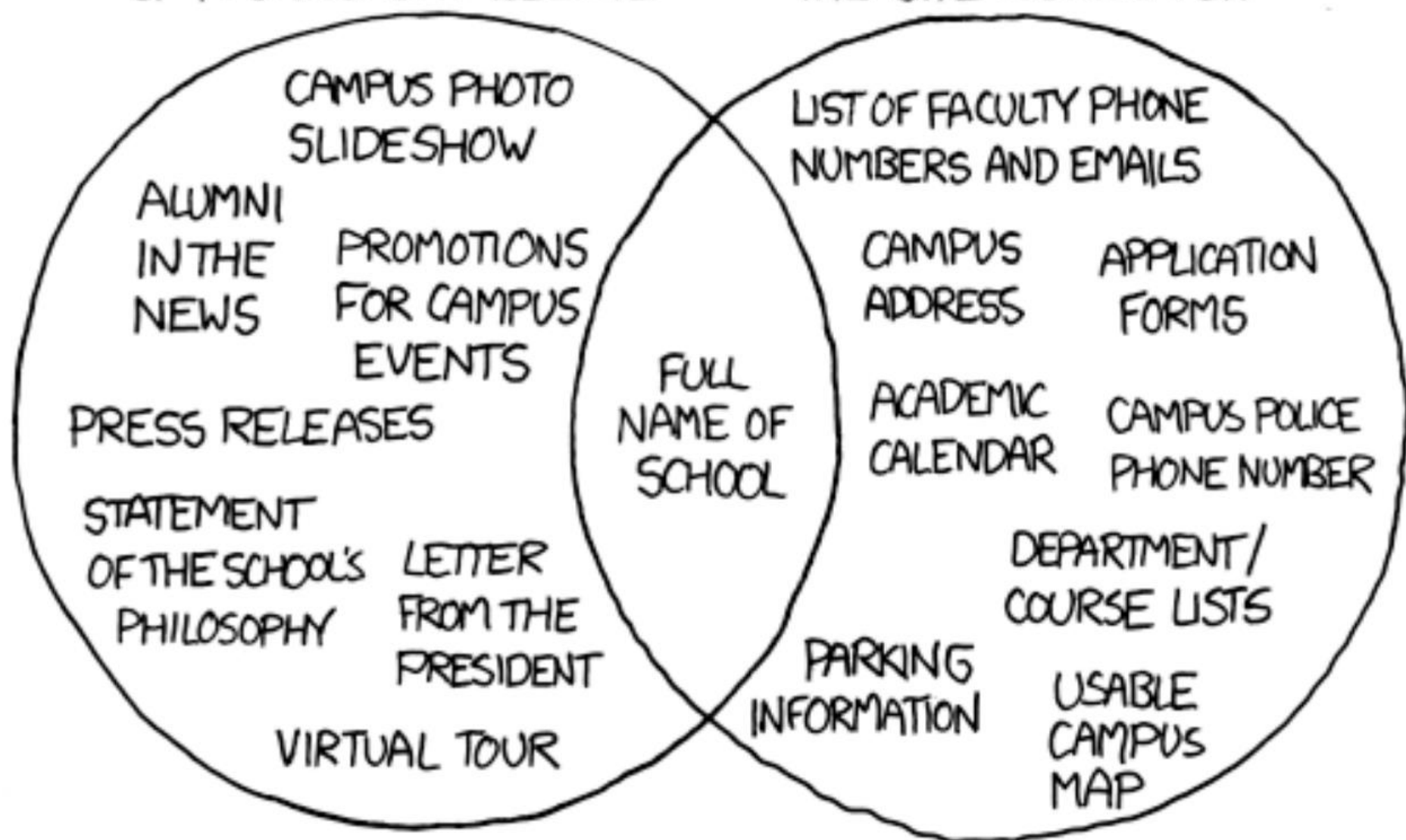
    <uses-sdk android:minSdkVersion="10" />

    <uses-permission android:name="android.permission.INTERNET" />

    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name" >
        <activity
            android:name=".HelloWebView"
            android:label="@string/app_name"
            android:theme="@android:style/Theme.NoTitleBar" >
```

THINGS ON THE FRONT PAGE
OF A UNIVERSITY WEBSITE

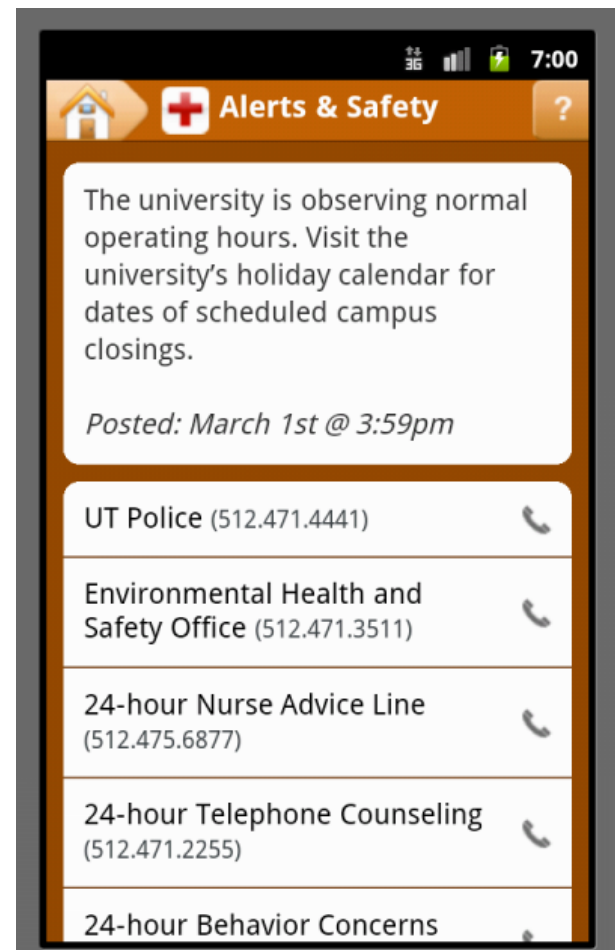
THINGS PEOPLE GO TO
THE SITE LOOKING FOR



Current Result



Clicking link actually leads to the default Android browser



Handling URL Requests

- To enable activity to handle its own URL requests create an inner class that extends WebViewClient

```
private class HelloWebViewClient extends WebViewClient {  
    @Override  
    public boolean shouldOverrideUrlLoading(WebView view,  
        String url) {  
        view.loadUrl(url);  
        return true;  
    }  
}
```

- set client for mWebView

```
mWebView.setWebViewClient(new HelloWebViewClient());
```

Navigating

- Making previous changes disables the back button
- Must override onKeyDown method
- Use WebView object to see if possible to go back

```
@Override
public boolean onKeyDown(int keyCode, KeyEvent event) {
    if ((keyCode == KeyEvent.KEYCODE_BACK)
        && mWebView.canGoBack()) {
        mWebView.goBack();
        return true;
    }
    return super.onKeyDown(keyCode, event);
}
```

Javascript

- If web page displayed in simple WebView contains JavaScript ...
- enable JavaScript:

```
WebView myWebView = (WebView) findViewById(R.id.webview)
WebSettings webSettings = myWebView.getSettings();
webSettings.setJavaScriptEnabled(true);
```

- create interface to act as bridge between JavaScript and Android
 - Example: convert JavaScript alert to Android Dialog

WEB SERVICES

Web Services

- "Web services are a means of exposing an API over a technology-neutral network endpoint."
- "They are a means to call a remote method or operation that's not tied to a specific platform or vendor and get a result."
 - Android in Action 3rd edition

Web Services Sources

- <http://www.programmableweb.com/apis/directory/1?sort=mashups>

API	Description	Category
Google Maps	Mapping services	Mapping
Twitter	Microblogging service	Social
YouTube	Video sharing and search	Video
Flickr	Photo sharing service	Photos
Amazon eCommerce	Online retailer	Shopping
Facebook	Social networking service	Social
Twilio	Telephony service	Telephony
eBay	eBay Search service	Search
Last.fm	Online radio service	Music
Google Search	Search services	Search
Microsoft Bing Maps	Mapping services	Mapping
Twilio SMS	SMS messaging service	Messaging
del.icio.us	Social bookmarking	Bookmarks
Yahoo Search	Search services	Search

Example: Flickr API

- <http://www.flickr.com/services/api/>

The App Garden

[Create an App](#) | [API Documentation](#) | [Feeds](#) | [What is the App Garden?](#)

The Flickr API is available for non-commercial use by outside developers. Commercial use is possible by prior arrangement.

Read these first:

- [Developer Guide](#)
- [Overview](#)
- [Encoding](#)
- [User Authentication](#)
- [Dates](#)
- [Tags](#)
- [URLs](#)
- [Buddyicons](#)
- [Flickr APIs Terms of Use](#)

API Methods

activity

- [flickr.activity.userComments](#)
- [flickr.activity.userPhotos](#)

auth

- [flickr.auth.checkToken](#)
- [flickr.auth.getFrob](#)
- [flickr.auth.getFullToken](#)
- [flickr.auth.getToken](#)

auth.oauth

Flickr API Methods

- create url using API method name and parameters, plus api key
- 3 request formats
 - REST
 - XML-RPC
 - SOAP
- Multiple Response Formats

API Methods

interestingness

- flickr.interestingness.getList

photos

- flickr.photos.addTags
- flickr.photos.delete
- flickr.photos.getAllContexts
- flickr.photos.getContactsPhotos
- flickr.photos.getContactsPublicPhotos
- flickr.photos.getContext
- flickr.photos.getCounts
- flickr.photos.getExif
- flickr.photos.getFavorites
- flickr.photos.getInfo
- flickr.photos.getNotInSet
- flickr.photos.getPerms
- flickr.photos.getRecent
- flickr.photos.getSizes
- flickr.photos.getUntagged
- flickr.photos.getWithGeoData
- flickr.photos.getWithoutGeoData
- flickr.photos.recentlyUpdated
- flickr.photos.removeTag
- flickr.photos.search

Flickr API Methods

flickr.photos.search

Return a list of photos matching some criteria. Only photos visible to the calling user will be returned. To return private or semi-private photos, the caller must be authenticated with 'read' permissions, and have permission to view the photos. Unauthenticated calls will only return public photos.

Authentication

This method does not require authentication.

Arguments

api_key (Required)

Your API application key. [See here](#) for more details.

user_id (Optional)

The NSID of the user whose photo to search. If this parameter isn't passed then everybody's public photos will be searched. A value of "me" will search against the calling user's photos for authenticated calls.

tags (Optional)

A comma-delimited list of tags. Photos with one or more of the tags listed will be returned. You can exclude results that match a term by prepending it with a - character.

Sample Request URL

- `https://api.flickr.com/services/rest/
?method=flickr.photos.search
&api_key=754a89ad04e0b72f42ffb77f41
2c021e
&tags=blue,cool,pretty`
- `https://api.flickr.com/services/rest/?met
hod=flickr.photos.search&api_key=754a8
9ad04e0b72f42ffb77f412c021e&tags=bl
ue,cool,pretty`

Result

```
-<rsp stat="ok">
- <photos page="1" pages="99433" perpage="100" total="9943226">
  <photo id="10925924743" owner="100272092@N07" secret="37d0e7af76" server="7455" farm="8" title="Dribblebabies Vendor Fair" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925643325" owner="100272092@N07" secret="016b025f56" server="3758" farm="4" title="Dribblebabies Vendor Fair" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925736264" owner="104102768@N06" secret="74a4bca1cf" server="7357" farm="8" title="Sky Blue Glass Evil Eye Genuine Mag Bracelet, #3" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925631155" owner="85498821@N08" secret="2af57f47e2" server="5475" farm="6" title="" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925746854" owner="29342764@N04" secret="98695778d3" server="2816" farm="3" title="Beaver Moon" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925932073" owner="100272092@N07" secret="55ab54525f" server="3828" farm="4" title="Dribblebabies Vendor Fair" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925625596" owner="85498821@N08" secret="1fa0e20d12" server="3832" farm="4" title="Up there" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="8059159748" owner="61674778@N07" secret="4e41dba87f" server="8457" farm="9" title="Jeannette, Haarlem 2011: Up and away" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925648603" owner="94210132@N08" secret="b5af6e53e3" server="3763" farm="4" title="franz bread delivery truck" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925850493" owner="104106325@N04" secret="8cbcd82b87" server="2822" farm="3" title="Beautiful 30 Strand Blue Seed Bead Length" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925479495" owner="15119982@N02" secret="0efcdd1441" server="5532" farm="6" title="MBlatchford-20131118-0216" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925523046" owner="35770825@N03" secret="bc195e7b6e" server="7322" farm="8" title="Otafukuma Pokupoku Stuffed Animal (み)" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925575225" owner="28099933@N03" secret="da3deed406" server="7305" farm="8" title="blue pencil" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925530496" owner="35770825@N03" secret="037ec18650" server="5500" farm="6" title="Otafukuma Pokupoku Stuffed Animal (み)" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10310695864" owner="27278265@N02" secret="be385d5711" server="3783" farm="4" title="Un temps pour voir la beauté de la nature" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925755163" owner="35770825@N03" secret="2dfe2baee8" server="3761" farm="4" title="Otafukuma Pokupoku Stuffed Animal (み)" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925625834" owner="86818001@N08" secret="35a6e46e9a" server="7304" farm="8" title="missing" ispublic="1" isfriend="0" isfamily="0"/>
  <photo id="10925611584" owner="35770825@N03" secret="bc84b0ed49" server="3675" farm="4" title="Otafukuma Pokupoku Stuffed Animal (み)" ispublic="1" isfriend="0" isfamily="0"/>
```

Parse Result to URL for Picture

Photo Source URLs

You can construct the source URL to a photo once you know its ID, server ID, farm ID and secret, as returned by the API.

The URL takes the following format:

```
http://farm{farm-id}.staticflickr.com/{server-id}/{id}_{secret}.jpg
or
http://farm{farm-id}.staticflickr.com/{server-id}/{id}_{secret}_{mstzb}.jpg
or
http://farm{farm-id}.staticflickr.com/{server-id}/{id}_{o-secret}_o.jpg
```

* Before November 18th, 2011 the API returned image URLs with hostnames like: "farm{farm-id}.static.flickr.com/{server-id}/{id}_{secret}.jpg"

Size Suffixes

The letter suffixes are as follows:

- s small square 75x75
- q large square 150x150
- t thumbnail, 100 on longest side
- m small, 240 on longest side

Photo URL and Result

```
<photo id="10925746854" owner="29342764@N04" secret="98695778d3" server="2816"  
farm="3" title="Beaver Moon" ispublic="1" isfriend="0" isfamily="0"/>
```

http://farm{farm-id}.staticflickr.com/{server-id}/{id}_{secret}.jpg
or

- http://farm3.staticflickr.com/2816/
10925746854_98695778d3.jpg
- http://farm3.staticflickr.com/2816/10925746
854_98695778d3.jpg

Flickr Example



JSON Format

- Flickr allows request for response format
 - one example, JSON

```
jsonFlickrApi({"photos":{"page":1, "pages":99436, "perpage":100,
"total":"9943595", "photo":[{"id":"8987334232",
"owner":"95960745@N05", "secret":"8ea7fa4884", "server":"3700",
"farm":4, "title":"ladder", "ispublic":1, "isfriend":0,
"isfamily":0}, {"id":"10922370846", "owner":"67877697@N07",
"secret":"9ac9663673", "server":"3670", "farm":4, "title":"Underneath
U.F.O.s", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"10926178565", "owner":"34128007@N04", "secret":"f4212fe642",
"server":"2875", "farm":3, "title":"Los Angeles City Hall",
"ispublic":1, "isfriend":0, "isfamily":0}, {"id":"10926310046",
"owner":"78618023@N04", "secret":"bc70fd4960", "server":"3678",
"farm":4, "title":"DSC_9235", "ispublic":1, "isfriend":0,
"isfamily":0}, {"id":"10926152545", "owner":"26870747@N04",
"secret":"80ab7da855", "server":"7328", "farm":8, "title":"Sunday
Sunrise", "ispublic":1, "isfriend":0, "isfamily":0},
{"id":"10926228834", "owner":"69774579@N07", "secret":"aaf5eeced1",
"server":"5533", "farm":6, "title":"Production of light",
```

WEATHER BUG EXAMPLE

WeatherBug Example

- From Deitel Android Programmers: An App-Driven Approach
- Example for Tablets
 - Fragments
 - tabbed navigation in Action Bar
 - Widget for home screen
- Our focus is on the use of Web Services

WeatherBug API

A promotional graphic for the WeatherBug API. It features a blue header with the text 'WeatherBug API'. Below this is a light blue bar with 'USA'. The main body has a dark blue background with a white grid pattern and a bright blue sunburst on the right. The text 'Build your own weather application using WeatherBug API tools' is prominently displayed. At the bottom, a white bar contains the text 'WeatherBug API lets novice to advanced developers create amazing weather applications using WeatherBug data.'



WeatherBug API

USA

**Build your own weather application
using WeatherBug API tools**

WeatherBug API lets novice to advanced developers create amazing weather applications using WeatherBug data.

WeatherView App - Current

 Weather Viewer Current Conditions Five Day Forecast  Add New City


Boston

Chicago

Dallas

Denver

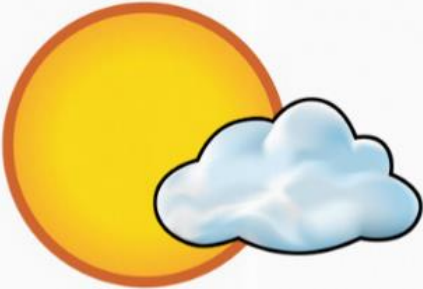
New York

Round Rock 

San Diego

San Francisco

Seattle



Round Rock TX, 78681 United States

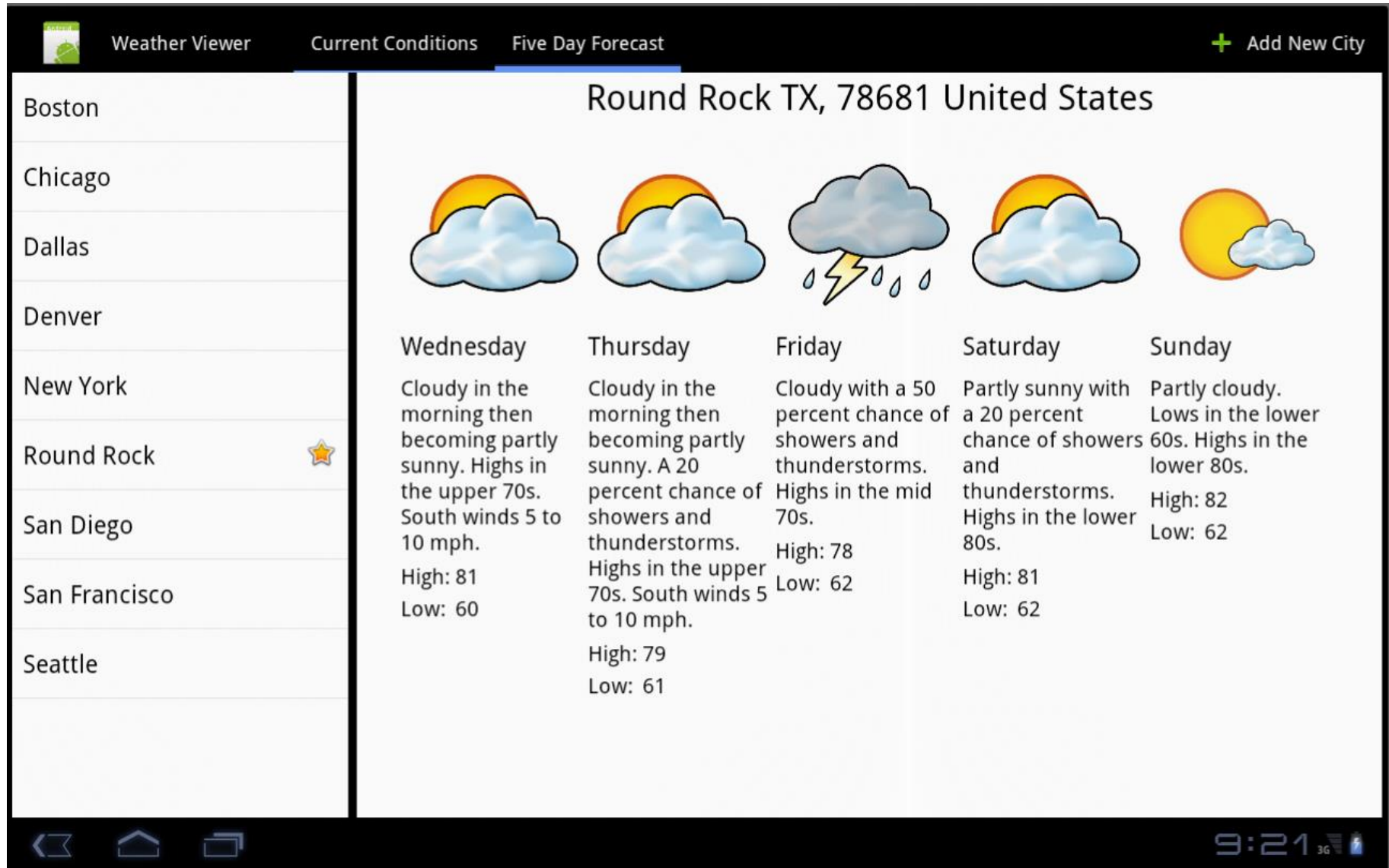
Temperature:
79°F

Feels like:
79°F

Humidity:
52%

Chance of Precipitation:
0%

WeatherView App - Five Day



Use of API

- most API's require registration and a key value
- key used in requests

```
// construct Weatherbug API URL
URL url = new URL(resources.getString(
    R.string.location_url_pre_zipcode) + zipcodeString +
    "&api_key=A5559065586");
```


WeatherBug Web Services

- Three classes deal with making requests via the WeatherBug API in WeatherView
- ReadLocationTask
 - use zip to get location information
- ReadForecastTask
 - read current forecast for given zip code
- ReadFiveDayForecastTask
 - get forecast for next five days for given zip

Tasks

- All three class use AsyncTask
 - chose to extend
- constructors
- override doInBackground method
- override onPostExecute method
- define their own listeners
- Keep the UI thread responsive by using AsyncTask to perform potentially slow tasks

AsyncTask

- "[AsyncTask](#) allows you to perform asynchronous work on your user interface. It performs the blocking operations in a worker thread and then publishes the results on the UI thread, without requiring you to handle threads and/or handlers yourself."
- Task started by invoking the execute method
- <http://developer.android.com/reference/android/os/AsyncTask.html>

ReadLocationTask

- Created with Context, zip code, and Listener
- Listener updated in postExecute method

```
public ReadLocationTask(String zipCodeString,  
    Context context,  
    LocationLoadedListener listener) {  
    this.zipcodeString = zipCodeString;  
    this.context = context;  
    this.resources = context.getResources();  
    this.weatherLocationLoadedListener = listener;  
}
```

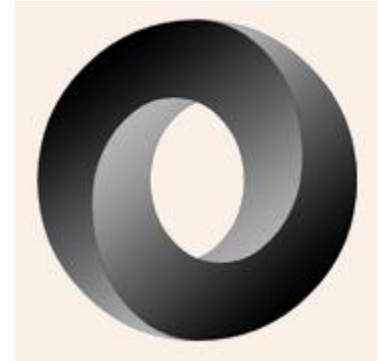
ReadLocationTask - doInBackground

- Creates URL using zip code and API key

```
// load city name in background thread
@Override
protected String doInBackground(Object... params)
{
    try {
        // construct Weatherbug API URL
        URL url = new URL(resources.getString(
            R.string.location_url_pre_zipcode) + zipcodeString +
            "&api_key=A5559065586");

        Reader forecastReader = new InputStreamReader(
            url.openStream());
    }
}
```

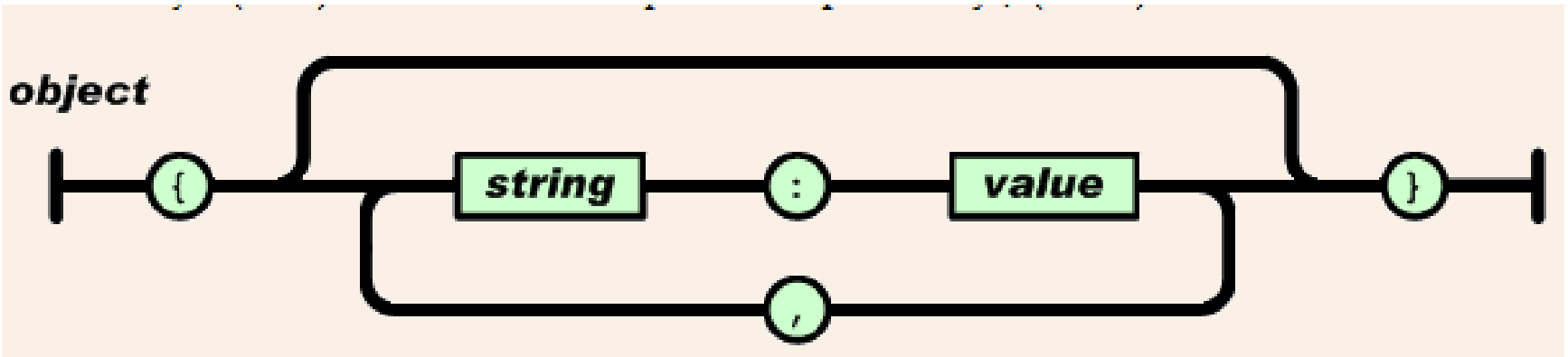
JSON



- JavaScript Object Notation
- a way to represent JavaScript objects as Strings
- alternative to XML for passing data between servers and clients
- designed for data interchange format that humans can also read and write

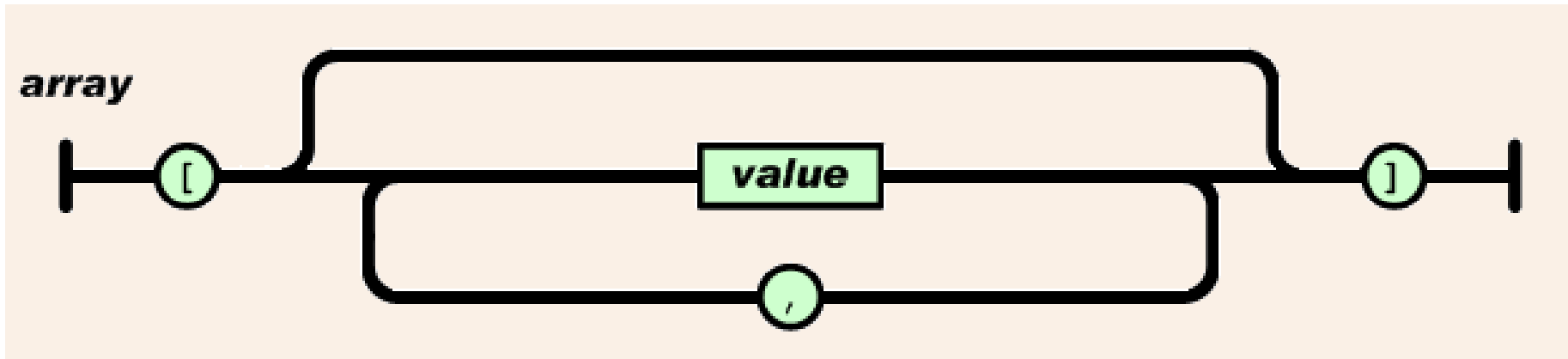
JSON Format

- Built on two structures
 - collection of name-value pairs: a.k.a. objects, records, structs, etc.
 - an ordered list of values: a.k.a. an array
- objects



JSON Format

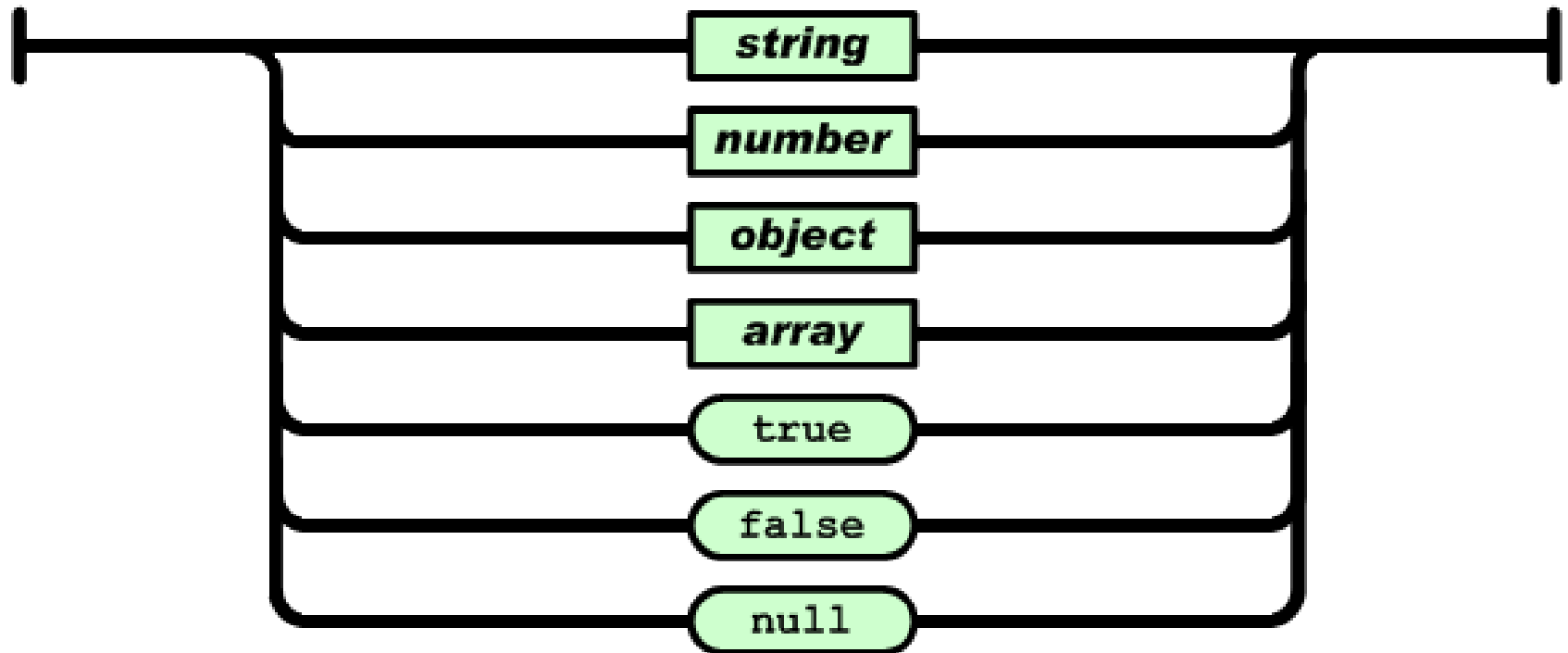
- arrays



- values
 - string, number, object, array, true, false, null

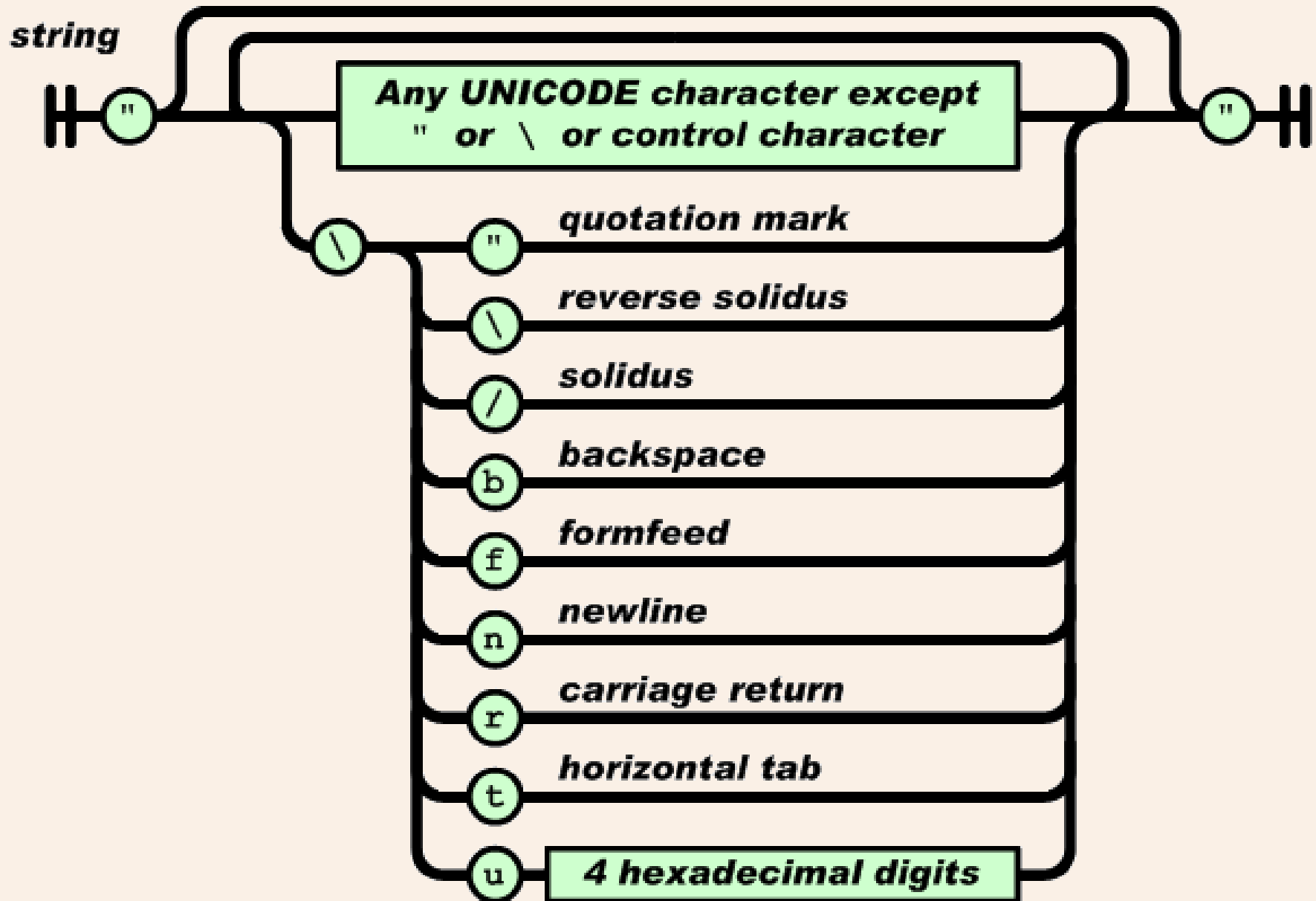
JSON Values

value

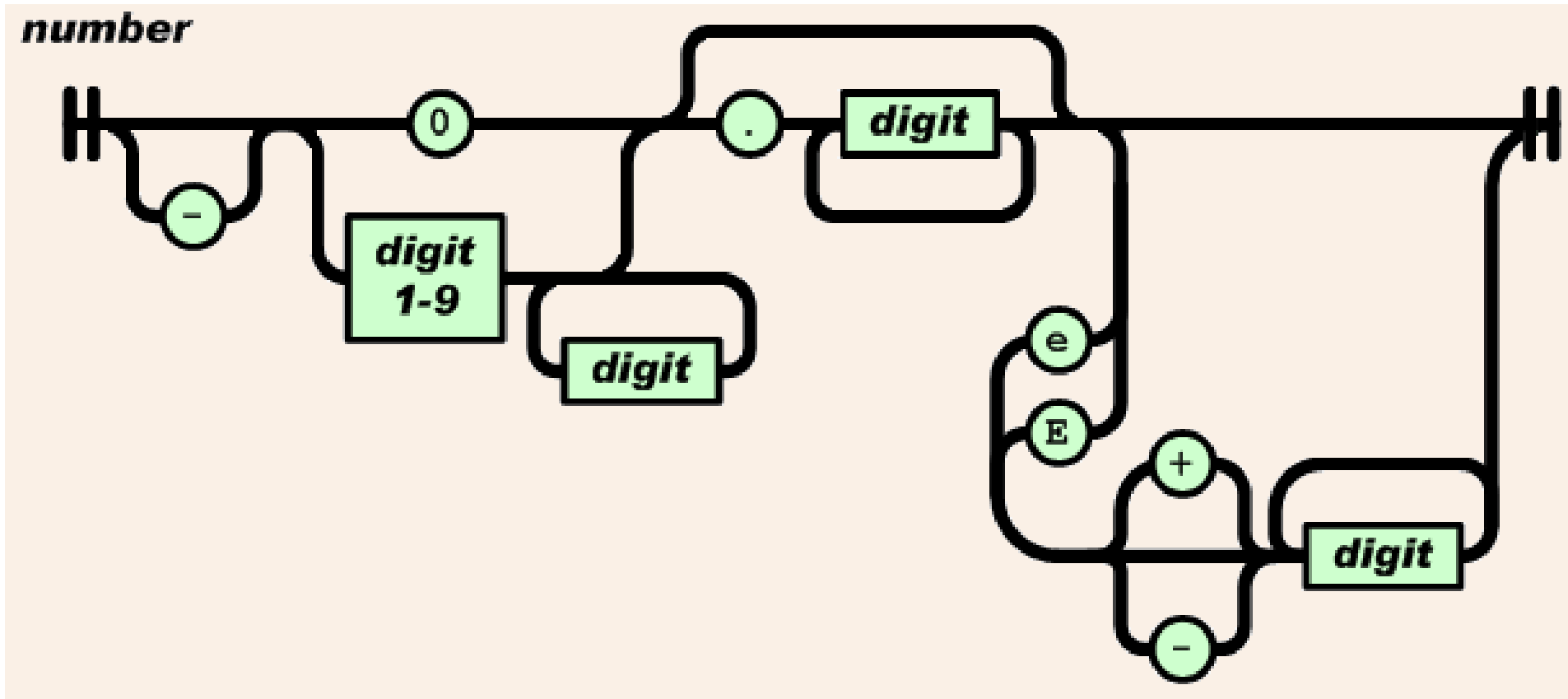


Syntax Diagrams for string and number: <http://www.json.org/>

JSON Strings



JSON Numbers



JSON Examples

- value (String):
 - "Round Rock"
- array:
 - ["Round Rock", "Dallas", "Houston"]
- object
 - {"height":70,"weight":165}

Results For ReadLocationTask

- `http://i.wxbug.net/REST/Direct/GetLocation.ashx?zip=78681&api_key=xxxXX`
 - where xxxxx is your API key
- Result:
 - `{"location": {"city": "Round Rock", "cityCode": null, "country": "United States", "dma": "635", "isUs": true, "lat": 30.5123, "lon": -97.7117, "state": "TX", "zipCode": "78681"}}`

Parsing JSON

- JsonReader class in Android API
- Read JSON encoded values as a stream of tokens
- ReadLocationTask uses a JsonReader to parse the JSON returned by the web request
- Pulls out city, state, and country string to display in View

Creating JsonReader

- and check given zip returns a valid location

```
JsonReader forecastJsonReader = new JsonReader(forecastReader);  
forecastJsonReader.beginObject(); // read the first Object
```

```
String name = forecastJsonReader洗洗洗Name();
```

```
// if the name indicates that the next item describes the  
// zipcode's location
```

```
if (name.equals(resources.getString(R.string.location))) {  
    forecastJsonReader.beginObject();  
    String洗洗洗洗洗String;
```

Reading Location Data

```
while (forecastJsonReader.hasNext()) {
    nextNameString = forecastJsonReader.nextName();
    // if the name indicates that the next item describes the
    // zipcode's corresponding city name
    if ((nextNameString).equals(
        resources.getString(R.string.city)))
        cityString = forecastJsonReader.nextString();
    else if ((nextNameString).equals(resources.
        getString(R.string.state)))
        stateString = forecastJsonReader.nextString();
    else if ((nextNameString).equals(resources.
        getString(R.string.country)))
        countryString = forecastJsonReader.nextString();
    else
        forecastJsonReader.skipValue();
}

forecastJsonReader.close();
```


onPostExecute

- Send the city, state, and country data to the listener

```
// executed back on the UI thread after the city name loads
protected void onPostExecute(String nameString) {
    if (cityString != null)
        weatherLocationLoadedListener.onLocationLoaded(cityString,
            stateString, countryString);
    else {
        Toast errorToast = Toast.makeText(context, resources.getString(
            R.string.invalid_zipcode_error), Toast.LENGTH_LONG);
        errorToast.setGravity(Gravity.CENTER, 0, 0);
        errorToast.show();
    }
}
```

ReadForecastTask

- Similar in nature to ReadLocationTask, but different url for different data
- {"forecastHourlyList":
[{"chancePrecip":"10","dateTime":1332882000000,
"desc":"PartlyCloudy","dewPoint":64,"feelsLike":73,
"feelsLikeLabel":"Heat Index","humidity":"74",
"icon":"cond002","skyCover":null,
"temperature":73,"windDir":null,"windSpeed":10},
{"chancePrecip":"10","dateTime":1332885600000,
"desc":"Partly Cloudy","dewPoint":64,"feelsLike":70,
"feelsLikeLabel":"Heat Index","humidity":"81",
"icon":"cond002","skyCover":null,
"temperature":70,"windDir":null,"windSpeed":11},
and on for another 158 hours

ReadForecastTask

- Also downloads image for current condition

```
// get the sky condition image Bitmap
public static Bitmap getIconBitmap(String conditionString,
    Resources resources, int bitmapSampleSize) {
    Bitmap iconBitmap = null;
    try {
        // create a URL pointing to the image on WeatherBug's site
        URL weatherURL = new URL(resources.getString(
            R.string.pre_condition_url) + conditionString +
            resources.getString(R.string.post_condition_url));

        Log.d(TAG, weatherURL.toString());

        BitmapFactory.Options options = new BitmapFactory.Options();
        if (bitmapSampleSize != -1)
            options.inSampleSize = bitmapSampleSize;

        // save the image as a Bitmap
        iconBitmap = BitmapFactory.decodeStream(weatherURL.
            openStream(), null, options);
    }
}
```

Icons Obtained From WeatherBug



ReadFiveDayForecastTask

- {"dateTime":1332892800000,
"dayDesc":"Partly Cloudy","dayIcon":"cond003",
"dayPred":"Cloudy in the morning...becoming partly
cloudy. Patchy fog in the morning. Highs 61 to 66. Light
winds becoming west 15 mph with gusts to 25 mph in
the afternoon.",
"dayTitle":"Wednesday","hasDay":true,
"hasNight":true,"high":"66","hourly":null,"low":"54","ni
ghtDesc":"Drizzle","nightIcon":"cond162",
"nightPred":"Partly cloudy in the evening...becoming
cloudy. Patchy fog and patchy drizzle overnight. Lows 49
to 55. Areas of winds northwest 15 to 20 mph with
gusts to 25 mph in the evening becoming light.",
"nightTitle":"WednesdayNight","title":"Wednesday"},

Displaying Data

- App does not try and display all data, just chooses "most important"
- icon
- day of week
- day prediction
- high temp
- low temp