How to parse freeform street/postal address out of text, and into components

Asked 12 years, 6 months ago Modified 2 months ago

Viewed 165k times

176

We do business largely in the United States and are trying to improve user experience by combining all the address fields into a single text area. But there are a few problems:

- The address the user types may not be correct or in a standard format
- The address must be separated into parts (street, city, state, etc.) to process credit card payments
- Users may enter more than just their address (like their name or company with it)
- Google can do this but the Terms of Service and query limits are prohibitive, especially on a tight budget

Apparently, this is a common question:

- PHP script to parse address?
- How do I parse the free format address to save into the DataBase

- java postal address parser
- More efficient way to extract address components
- How can i show a pre populated postal address in contacts screen with street, city, zip on android
- PHP regexp US address

Is there a way to isolate an address from the text around it and break it into pieces? Is there a regular expression to parse addresses?

parsing street-address

Share Follow

edited Jan 19, 2022 at 14:17
General Grievance
4,977 ● 37 ● 36 ● 54

asked Jun 22, 2012 at 16:19
Matt
23.7k ● 18 ● 73 ● 115

6 Answers

Sorted by: Highest score (default)

341

I saw this question a lot when I worked for an address verification company. I'm posting the answer here to make it more accessible to programmers who are searching around with the same question. The company I was at processed billions of addresses, and we learned a lot in the process.

First, we need to understand a few things about addresses.

Addresses are not regular

This means that regular expressions are out. I've seen it all, from simple regular expressions that match addresses in a very specific format, to this:

> /\s+(\d{2,5}\s+)(?![a|p]m\b)(([a-zA-Z|\s+]{1,5}){1,2})?([\s|,|.]+)?(([a-zA-Z|\s+]{1,30}){1,4})(court|ct|street|st|drive|dr|lane|ln|road|rd|blvd)([\s|,|.|;]+)?(([a-zA-Z|\s+]{1,30}){1,2})([\s|,|.]+)?\b(AK|AL|AR|AZ|CA|CO|CT|DC|DE|FL|GA|GU|HI|IA|ID|IL|IN|KS|KY|LA|MA|MD|ME|MI|MN|MO|MS|MT|NC|ND|NE|NH|NJ|NM|NV|NY|OH|OK|OR|PA|RI|SC|SD|TN|TX|UT|VA|VI|VT|WA|WI|WV|WY)([\s|,|.]+)?(\s+\d{5})?([\s|,|.]+)/i

... to this where a 900+ line-class file generates a supermassive regular expression on the fly to match even more. I don't recommend these (for example, here's a fiddle of the above regex, that makes plenty of mistakes). There isn't an easy magic formula to get this to work. In theory and *by* theory, it's not possible to match addresses with a regular expression.

USPS Publication 28 documents the many formats of addresses that are possible, with all their keywords and variations. Worst of all, addresses are often ambiguous. Words can mean more than one thing ("St" can be "Saint" or "Street") and there are words that I'm pretty sure they invented. (Who knew that "Stravenue" was a street suffix?)

You'd need some code that really understands addresses, and if that code does exist, it's a trade secret. But you could probably roll your own if you're really into that.

Addresses come in unexpected shapes and sizes

Here are some contrived (but complete) addresses:

```
1)  102 main street
    Anytown, state

2)  400n 600e #2, 52173

3)  p.o. #104 60203
```

Even these are possibly valid:

```
4)  829 LKSDFJlkjsdflkjsdljf Bkpw 12345

5)  205 1105 14 90210
```

Obviously, these are not standardized. Punctuation and line breaks not guaranteed. Here's what's going on:

1. **Number 1** is complete because it contains a street address and a city and state. With that information, there's enough to identify the address, and it can be considered "deliverable" (with some standardization).
2. **Number 2** is complete because it also contains a street address (with secondary/unit number) and a 5-digit ZIP code, which is enough to identify an address.
3. **Number 3** is a complete post office box format, as it contains a ZIP code.
4. **Number 4** is also complete because the ZIP code is unique, meaning that a private entity or corporation has purchased that address space. A unique ZIP code is for high-volume or concentrated delivery spaces. Anything addressed to ZIP code 12345 goes to General Electric in Schenectady, NY. This example won't reach anyone in particular, but the USPS would still be able to deliver it.
5. **Number 5** is also complete, believe it or not. With just those numbers, the full address can be discovered when parsed against a database of all possible addresses. Filling in the missing directionals, secondary designator, and ZIP+4 code is trivial when you see each number as a component. Here's what it looks like, fully expanded and standardized:

> 205 N 1105 W Apt 14

> Beverly Hills CA 90210-5221

Address data is not your own

In most countries that provide official address data to licensed vendors, the address data itself belongs to the governing agency. In the US, the USPS owns the addresses. The same is true for Canada Post, Royal Mail, and others, though each country enforces or defines ownership a little differently. Knowing this is important, since it usually forbids reverse-engineering the address database. You have to be careful how to acquire, store, and use the data.

Google Maps is a common go-to for quick address fixes, but the TOS is rather prohibitive; for example, you can't use their data or APIs without showing a Google Map, and for non-commercial purposes only (unless you pay), and you can't store the data (except for temporary caching). Makes sense. Google's data is some of the best in the world. However, Google Maps does *not* verify the address. If an address does not exist, it will still show you where the address *would* be if it *did* exist (try it on your own street; use a house number that you know doesn't exist). This is useful sometimes, but be aware of that.

Nominatim's usage policy is similarly limiting, especially for high volume and commercial use, and the data is mostly drawn from free sources, so it isn't as well maintained (such is the nature of open projects) -- however, this may still suit your needs. It is supported by a great community.

The USPS itself has an API, but it goes down a lot and comes with no guarantees nor support. It might also be hard to use. Some people use it sparingly with no problems. But it's easy to miss that the USPS requires that you use their API only for confirming addresses to ship through them.

People expect addresses to be hard

Unfortunately, we've conditioned our society to expect addresses to be complicated. There's dozens of good UX articles all over the Internet about this, but the fact is, if you have an address form with individual fields, that's what users expect, even though it makes it harder for edge-case addresses that don't fit the format the form is expecting, or maybe the form requires a field it shouldn't. Or users don't know where to put a certain part of their address.

I could go on and on about the bad UX of checkout forms these days, but instead I'll just say that combining the addresses into a single field will be a *welcome* change -- people will be able to type their address how they see fit, rather than trying to figure out your lengthy form.

However, this change will be *unexpected* and users may find it a little jarring at first. Just be aware of that.

Part of this pain can be alleviated by putting the country field out front, before the address. When they fill out the country field first, you know how to make your form appear. Maybe you have a good way to deal with single-field US addresses, so if they select United States, you can reduce your form to a single field, otherwise show the component fields. Just things to think about!

Now we know why it's hard; what can you do about it?

The USPS licenses vendors through a process called CASS™ Certification to provide verified addresses to customers. These vendors have access to the USPS database, updated monthly. Their software must conform to rigorous standards to be certified, and they don't often require agreement to such limiting terms as discussed above.

There are many CASS-Certified companies that can process lists or have APIs: Melissa Data, Experian QAS, and SmartyStreets to name a few.

(Due to getting flak for "advertising" I've truncated my answer at this point. It's up to you to find a solution that works for you.)

The Truth: Really, folks, I don't work at any of these companies. It's not an advertisement.

Share Follow

edited Feb 25, 2023 at 8:13

Ry- ♦
225k ● 56 ● 488 ● 496

answered Jun 22, 2012 at 16:19

Matt
23.7k ● 18 ● 73 ● 115

---

q: when you were working on CASS compliance... just guestimating by memory, how much of the code is algorithm based (anything from regex to ML) versus data driven (state/zip/road lookups)? – Scott Brickey Aug 5, 2017 at 13:36

---

1 @ScottBrickey Hard to say. Here's [the technical manual]() for implementing CASS-certified software. Ultimately they don't care *how* you do it as long as it passes the tests. It's very algorithm heavy, but there are also lots of data lookups. – Matt Aug 6, 2017 at 5:46

@Matt thanks... i've looked at their docs a few times... a lot of it appears to be the order of lookups (validate zip vs city/state)... sprinkled in with a bit of fuzzy text matching (levenshtein seems to be a common approach)... but I keep figuring that it can't be that direct and that I must be missing aspects. – Scott Brickey Aug 7, 2017 at 1:11

---

2 @KamalHussain He wasn't suggesting that the regex *did* work for all cases. Two sentences later he says: "I don't recommend these…" – Michael come lately Mar 18, 2018 at 3:54

---

1 I programmed assembler and C for a large demographic marketing/direct mailer through the early 90s and I remember

how valuable were the capabilities to standardize/complete addresses and to qualify for discounts by precisely coding down to the carrier. – grantwparks Apr 5, 2018 at 2:19

15

There are many street address parsers. They come in two basic flavors - ones that have databases of place names and street names, and ones that don't.

A regular expression street address parser can get up to about a 95% success rate without much trouble. Then you start hitting the unusual cases. The Perl one in CPAN, "Geo::StreetAddress::US", is about that good. There are Python and Javascript ports of that, all open source. I have an improved version in Python which moves the success rate up slightly by handling more cases. To get the last 3% right, though, you need databases to help with disambiguation.

A database with 3-digit ZIP codes and US state names and abbreviations is a big help. When a parser sees a consistent postal code and state name, it can start to lock on to the format. This works very well for the US and UK.

Proper street address parsing starts from the end and works backwards. That's how the USPS systems do it. Addresses are least ambiguous at the end, where country names, city names, and postal codes are relatively easy to recognize. Street names can usually be isolated. Locations on streets are the most complex to parse; there you encounter things such as "Fifth Floor" and "Staples Pavillion". That's when a database is a big help.

Share Follow

answered Apr 5, 2015 at 5:25

John Nagle

1,568 ● 1 ● 14 ● 16

There is also the CPAN module Lingua:EN::AddressParse. While slower than "Geo::StreetAddress::US, it gives a higher success rate. – Kim Ryan Jun 29, 2016 at 0:16

Could you recommend one without database for UK? thanks! – rex Jan 13, 2023 at 9:28

9

UPDATE: Geocode.xyz now works worldwide. For examples see https://geocode.xyz

For USA, Mexico and Canada, see geocoder.ca.

For example:

> Input: [something going on near the intersection of main and arthur kill rd new york]()
>
> Output:
>
> ```
> <geodata>
>   <latt>40.5123510000</latt>
>   <longt>-74.2500500000</longt>
>   <AreaCode>347,718</AreaCode>
>   <TimeZone>America/New_York</TimeZone>
>   <standard>
>     <street1>main</street1>
>     <street2>arthur kill</street2>
>     <stnumber/>
>     <staddress/>
>     <city>STATEN ISLAND</city>
> ```

>```
>     <prov>NY</prov>
>     <postal>11385</postal>
>     <confidence>0.9</confidence>
>   </standard>
> </geodata>
>```

You may also check the results in the web interface or get output as Json or Jsonp. eg. [I'm looking for restaurants around 123 Main Street, New York]()

Share Follow

edited Apr 25, 2018 at 20:29
Michael come lately
9,293 ● 7 ● 66 ● 92

answered Dec 22, 2015 at 0:41
Ervin Ruci
919 ● 7 ● 13

How you implemented the address parsing system using openaddress ? Are you using brute force strategy ? – user1112259 Jul 19, 2016 at 14:20

---

1 What do you mean by 'brute force'? Breaking up text into all possible combinations of possible address strings and comparing each one against a database of addresses is not practical and will take way more time to provide an answer than this system does. Openaddresses are one of the data sources for building a 'training set' of address formats for the algorithm. It uses this information to parse addresses out of unstructured text. – Ervin Ruci Jul 20, 2016 at 15:25

---

3 Another similar system is Geo::libpostal ([perltricks.com/article/announcing-geo--libpostal]()) They also use openstreetmap and openaddresses it seems, to build address templates on the fly – Ervin Ruci Jul 20, 2016 at 15:34

1 I just tested geocode.xyz's geoparser (send in text, get back location) on hundreds of actual addresses. Given a side by side with google map's API, and a global set of addresses, `geocode.xyz` 's `scantext` method failed most of the time. It always chose "Geneva,US" over "Geneva, Switzerland" and was generally US biased. – Marc Maxmeister Dec 4, 2018 at 16:29

It depends on the context. geocode.xyz/?scantext=Geneva,%20Switzerland will produce: Match Location Geneva, Switzerland, CH Confidence Score: 0.8 while geocode.xyz/?scantext=Geneva,%20USA will produce Match Location Geneva,US Confidence Score: 1.0 Also, you can region bias as follows: geocode.xyz/?scantext=Geneva,%20USA&region=CH – Ervin Ruci Dec 5, 2018 at 17:04

6

Here is a simple JavaScript address parser. It's pretty awful for every single reason that Matt gives in his dissertation above (which I almost 100% agree with: addresses are complex types, and humans make mistakes; better to outsource and automate this - when you can afford to).

But I decided to try.

This code works OK for parsing most Esri results for `findAddressCandidate` and also with some other (reverse)geocoders that return single-line address where street/city/state are delimited by commas. You can extend if you want or write country-specific parsers. Or just use this as case study of how challenging this exercise can be or at how lousy I am at JavaScript. I admit I only spent about thirty mins on this (future iterations could add caches, zip validation, and state lookups as well as user location context), but it worked for my use case: End user sees form that parses geocode search response into 4 textboxes. If address parsing comes out wrong (which is rare unless source data was poor) it's no big deal - the user gets to verify and fix it! (But for automated solutions could either discard/ignore or flag as error so dev can either support the new format or fix source data.)

```
/*
address assumptions:
- US addresses only (probably want separate parser f
- No country code expected.
- if last token is a number it is probably a postal
-- 5 digit number means more likely
- if last token is a hyphenated string it might be a
-- if both sides are numeric, and in form #####-####
- if city is supplied, state will also be supplied (
- zip/postal code may be omitted even if has city &
- state may be two-char code or may be full state na
- commas:
-- last comma is usually city/state separator
-- second-to-last comma is possibly street/city sepa
-- other commas are building-specific stuff that I d
- token count:
-- because units, street names, and city names may c
highly variable.
-- simplest address has at least two tokens: 714 OAK
-- common simple address has at least four tokens: 7
-- common full (mailing) address has at least 5-7:
--- 714 OAK, RUMTOWN, VA 59201
--- 714 S OAK ST, RUMTOWN, VA 59201
-- complex address may have a dozen or more:
--- MAGICICIAN SUPPLY, LLC, UNIT 213A, MAGIC TOWN MA
LAND OF MAGIC, MA 73122-3412
*/

var rawtext = $("textarea").val();
var rawlist = rawtext.split("\n");

function ParseAddressEsri(singleLineaddressString) {
  var address = {
    street: "",
    city: "",
    state: "",
    postalCode: ""
  };

  // tokenize by space (retain commas in tokens)
  var tokens = singleLineaddressString.split(/[\s]+/
  var tokenCount = tokens.length;
  var lastToken = tokens.pop();
  if (
    // if numeric assume postal code (ignore length,
    !isNaN(lastToken) ||
    // if hyphenated assume long zip code, ignore wh
    lastToken.split("-").length - 1 === 1) {
    address.postalCode = lastToken;
    lastToken = tokens.pop();
  }

  if (lastToken && isNaN(lastToken)) {
    if (address.postalCode.length && lastToken.lengt
      // assume state/province code ONLY if had post
      // otherwise it could be a simple address like
      // where "ST" for "street" looks like two-lett
      // possibly this could be resolved with regist
but meh. (and may collide anyway)
      address.state = lastToken;
      lastToken = tokens.pop();
    }
    if (address.state.length === 0) {
      // check for special case: might have State nar
      var stateNameParts = [lastToken.endsWith(",")
lastToken.length - 1) : lastToken];

      // check remaining tokens from right-to-left f
      while (2 + 2 != 5) {
        lastToken = tokens.pop();
        if (!lastToken) break;
        else if (lastToken.endsWith(",")) {
          // found separator, ignore stuff on left s
```

```
            tokens.push(lastToken); // put it back
            break;
          } else {
            stateNameParts.unshift(lastToken);
          }
        }
        address.state = stateNameParts.join(' ');
        lastToken = tokens.pop();
      }
    }

    if (lastToken) {
      // here is where it gets trickier:
      if (address.state.length) {
        // if there is a state, then assume there is a
        // PROBLEM: city may be multiple words (spaces
        // but we can pretty safely assume next-from-l
of the city name
        // most cities are single-name. It would be ve
context, like
        // the name of the city user is in. But ignore
        // ideally would have zip code service or look
the zip code.
        var cityNameParts = [lastToken.endsWith(",") ?
lastToken.length - 1) : lastToken];

        // assumption / RULE: street and city must hav
        // addresses that do not follow this rule will
space
        // but don't care because Esri formats put com
        var streetNameParts = [];

        // check remaining tokens from right-to-left f
        while (2 + 2 != 5) {
          lastToken = tokens.pop();
          if (!lastToken) break;
          else if (lastToken.endsWith(",")) {
            // found end of street address (may includ
care right now)
            // add token back to end, but remove trail
            tokens.push(lastToken.endsWith(",") ? last
lastToken.length - 1) : lastToken);
            streetNameParts = tokens;
            break;
          } else {
            cityNameParts.unshift(lastToken);
          }
        }
        address.city = cityNameParts.join(' ');
        address.street = streetNameParts.join(' ');
      } else {
        // if there is NO state, then assume there is
(easy)
        // reasoning: city names are not unique (Portl
so if user wants city they need to store state also
in Portland, OR, you don't care about city/state)
        // put last token back in list, then rejoin on
        tokens.push(lastToken);
        address.street = tokens.join(' ');
      }
    }
    // when parsing right-to-left hard to know if stre
city/state
    // hack fix for now is to shift stuff around.
    // assumption/requirement: will always have at lea
never just get "city, state"
    // could possibly tweak this with options or more
parsing&sniffing
    if (!address.city && address.state) {
      address.city = address.state;
      address.state = '';
    }
    if (!address.street) {
      address.street = address.city;
      address.city = '';
    }

    return address;
}

// get list of objects with discrete address propert
var addresses = rawlist
    .filter(function(o) {
      return o.length > 0
    })
    .map(ParseAddressEsri);
```

```
$("#output").text(JSON.stringify(addresses));
console.log(addresses);
```

```
<script src="https://ajax.googleapis.com/ajax/libs/j
</script>
<textarea>
27488 Stanford Ave, Bowden, North Dakota
380 New York St, Redlands, CA 92373
13212 E SPRAGUE AVE, FAIR VALLEY, MD 99201
1005 N Gravenstein Highway, Sebastopol CA 95472
A. P. Croll &amp; Son 2299 Lewes-Georgetown Hwy, Geo
11522 Shawnee Road, Greenwood, DE 19950
144 Kings Highway, S.W. Dover, DE 19901
Intergrated Const. Services 2 Penns Way Suite 405, N
Humes Realty 33 Bridle Ridge Court, Lewes, DE 19958
Nichols Excavation 2742 Pulaski Hwy, Newark, DE 1971
2284 Bryn Zion Road, Smyrna, DE 19904
VEI Dover Crossroads, LLC 1500 Serpentine Road, Suit
580 North Dupont Highway, Dover, DE 19901
P.O. Box 778, Dover, DE 19903
714 S OAK ST
714 S OAK ST, RUM TOWN, VA, 99201
3142 E SPRAGUE AVE, WHISKEY VALLEY, WA 99281
27488 Stanford Ave, Bowden, North Dakota
380 New York St, Redlands, CA 92373
10 Downing Street, Westminster, London SW1A 2AA, UK
P.O. Box 1234, Victoria, Seychelles
1 Chome-1-2 Oshiage, Sumida City, Tokyo 131-8588, Ja
Via Roma, 150, 00184 Roma RM, Italy
Karl Johans gate 22, 0159 Oslo, Norway
Sheikh Mohammed bin Rashid Blvd, Downtown Dubai, Dub
Paseo de la Castellana, 110, 28046 Madrid, Spain
Kurfürstendamm 26a, 10719 Berlin, Germany
Rue du Faubourg Saint-Honoré, 75008 Paris, France
Avenida Paulista, 1578, São Paulo, SP, 01310-200, Br
765 Pablo Neruda, Providencia, Santiago, 7501015, Ch
100 Queen St, Melbourne VIC 3000, Australia
4 Central Road, Ikoyi, Lagos, Nigeria
Tughlak Road Area, New Delhi, Delhi 110011, India
375 M.3 Donkaew, Amphoe Mae Rim, Chiang Mai 50180, T
21/F, 1 Hysan Avenue, Causeway Bay, Hong Kong
Stolitsa, 2011 Sofia, Bulgaria
```

```
Karlavägen 94, 115 26 Stockholm, Sweden
220 Bourke St, Melbourne VIC 3000, Australia
Ulitsa Bol'shaya Lubyanka, 26c1, Moscow, Russia
</textarea>
<div id="output">
</div>
```

Run code snippet Expand snippet

Share Follow

edited Oct 2 at 18:30

answered Mar 15, 2018 at 18:59

nothingisnecessary
6,197 ● 37 ● 63

disclaimer: my clients own their address data and run their own Esri servers. If you grab data from google, OSM, ArcGisOnline, or wherever, make sure it is OK to store and use it (many services have restrictions on how you can store, and for how long) – nothingisnecessary Mar 15, 2018 at 19:08

The first answer above makes a compelling case that this problem is unsolvable with regexes if you're dealing with a global address list. 200 countries have too many exceptions. In my testing, you can determine the country from a string rather reliably, then look up a specifc regex for each country - which is probably how the better APIs work.
– Marc Maxmeister Dec 4, 2018 at 16:31

3

For U.S. address parsing, I prefer using [the `usaddress` package](#) that is [available in `pip`](#).

```
python3 -m pip install usaddress
```

Usage sample:

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-

# address_parser.py
import sys
from usaddress import tag
from json import dumps, loads

if __name__ == '__main__':
    tag_mapping = {
        'Recipient': 'recipient',
        'AddressNumber': 'addressStreet',
        'AddressNumberPrefix': 'addressStreet',
        'AddressNumberSuffix': 'addressStreet',
        'StreetName': 'addressStreet',
        'StreetNamePreDirectional': 'addressStreet',
        'StreetNamePreModifier': 'addressStreet',
        'StreetNamePreType': 'addressStreet',
        'StreetNamePostDirectional': 'addressStreet',
        'StreetNamePostModifier': 'addressStreet',
        'StreetNamePostType': 'addressStreet',
        'CornerOf': 'addressStreet',
        'IntersectionSeparator': 'addressStreet',
        'LandmarkName': 'addressStreet',
        'USPSBoxGroupID': 'addressStreet',
        'USPSBoxGroupType': 'addressStreet',
        'USPSBoxID': 'addressStreet',
        'USPSBoxType': 'addressStreet',
        'BuildingName': 'addressStreet',
        'OccupancyType': 'addressStreet',
        'OccupancyIdentifier': 'addressStreet',
        'SubaddressIdentifier': 'addressStreet',
        'SubaddressType': 'addressStreet',
        'PlaceName': 'addressCity',
        'StateName': 'addressState',
        'ZipCode': 'addressPostalCode',
    }
    try:
        address, _ = tag(' '.join(sys.argv[1:]), tag_m
    except:
        with open('failed_address.txt', 'a') as fp:
            fp.write(sys.argv[1] + '\n')
        print(dumps({}))
    else:
        print(dumps(dict(address)))
```

Running `address_parser.py`:

```
python3 address_parser.py 9757 East Arcadia Ave. Saugu
{"addressStreet": "9757 East Arcadia Ave.", "addressCi
"addressState": "MA", "addressPostalCode": "01906"}
```

Share Follow

edited Dec 31, 2021 at 16:33
Michael come lately
9,293 ● 7 ● 66 ● 92

answered Jun 4, 2019 at 23:27
theBuzzyCoder
2,880 ● 2 ● 32 ● 26

1

I'm late to the party, but here is an Excel VBA script I wrote years ago for Australia. It can be easily modified to support other Countries. I've made a GitHub repository of the C# code here. I've hosted it on my site and you can download it here: http://jeremythompson.net/Rocks/ParseAddress.xlsm

Strategy

For any country with a PostCode that's numeric or can be matched with a RegEx my strategy works very well:

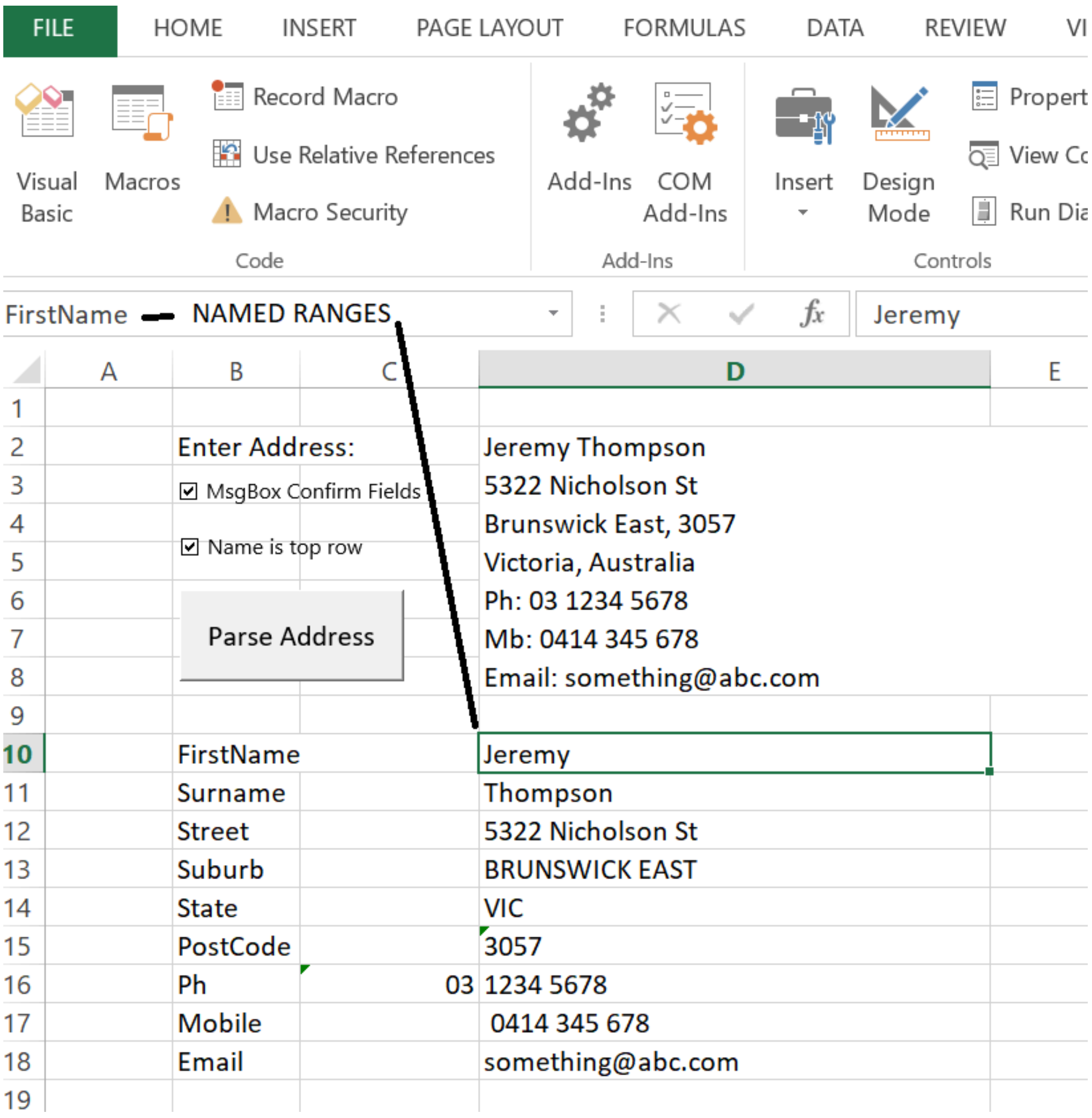
1. First we detect the First and Surname which are assumed to be the top line. Its easy to skip the name and start with the address by unticking the checkbox (called 'Name is top row' as shown below).
2. **Next its safe to expect the Address consisting of the Street and Number come before the Suburb and the St, Pde, Ave, Av, Rd, Cres, loop, etc is a separator.**
3. Detecting the Suburb vs the State and even Country can trick the most sophisticated parsers as there can be conflicts. To overcome this I use a PostCode look up based on the fact that after stripping Street and Apartment/Unit numbers as well as the PoBox,Ph,*Fax*,Mobile etc, only the PostCode number will remain. This is easy to match with a regEx to then look up the suburb(s) and country.

   > Your National Post Office Service will provide a list of post codes with Suburbs and States free of charge that you can store in an excel sheet, db table, text/json/xml file, etc.

4. Finally, since some Post Codes have multiple Suburbs we check which suburb appears in the

Address.

Example



	A	B	C	D	E
1					
2		Enter Address:		Jeremy Thompson	
3				5322 Nicholson St	
4				Brunswick East, 3057	
5				Victoria, Australia	
6				Ph: 03 1234 5678	
7				Mb: 0414 345 678	
8				Email: something@abc.com	
9					
10		FirstName		Jeremy	
11		Surname		Thompson	
12		Street		5322 Nicholson St	
13		Suburb		BRUNSWICK EAST	
14		State		VIC	
15		PostCode		3057	
16		Ph	03	1234 5678	
17		Mobile		0414 345 678	
18		Email		something@abc.com	
19					

VBA Code

DISCLAIMER, I know this code is not perfect, or even written well however its very easy to convert to any programming language and run in any type of application.

The strategy is the answer depending on your country and rules, take this code as an example:

```
Option Explicit

Private Const TopRow As Integer = 0

Public Sub ParseAddress()
Dim strArr() As String
Dim sigRow() As String
Dim i As Integer
Dim j As Integer
Dim k As Integer
Dim Stat As String
Dim SpaceInName As Integer
Dim Temp As String
Dim PhExt As String

On Error Resume Next

Temp = ActiveSheet.Range("Address")

'Split info into array
strArr = Split(Temp, vbLf)

'Trim the array
For i = 0 To UBound(strArr)
strArr(i) = VBA.Trim(strArr(i))
Next i

'Remove empty items/rows
ReDim sigRow(LBound(strArr) To UBound(strArr))
For i = LBound(strArr) To UBound(strArr)
    If Trim(strArr(i)) <> "" Then
        sigRow(j) = strArr(i)
        j = j + 1
    End If
Next i
ReDim Preserve sigRow(LBound(strArr) To j)

'Find the name (MUST BE ON THE FIRST ROW UNLESS CHECKB
i = TopRow
```

```
If ActiveSheet.Shapes("chkFirst").ControlFormat.Value

SpaceInName = InStr(1, sigRow(i), " ", vbTextCompare)

If ActiveSheet.Shapes("chkConfirm").ControlFormat.Valu
ActiveSheet.Range("FirstName") = VBA.Left(sigRow(i), S
Else
 If MsgBox("First Name: " & VBA.Mid$(sigRow(i), 1, Spa
vbYesNo, "Confirm Details") = vbYes Then ActiveSheet.R
VBA.Left(sigRow(i), SpaceInName)
End If

If ActiveSheet.Shapes("chkConfirm").ControlFormat.Valu
ActiveSheet.Range("Surname") = VBA.Mid(sigRow(i), Spac
Else
  If MsgBox("Surame: " & VBA.Mid(sigRow(i), SpaceInNam
vbYesNo, "Confirm Details") = vbYes Then ActiveSheet.R
VBA.Mid(sigRow(i), SpaceInName + 2)
End If
sigRow(i) = ""
End If

'Find the Street by looking for a "St, Pde, Ave, Av, R
For i = 1 To UBound(sigRow)
If Len(sigRow(i)) > 0 Then
    For j = 0 To 8
    If InStr(1, VBA.UCase(sigRow(i)), Street(j), vbTex

    'Find the position of the street in order to get t
    SpaceInName = InStr(1, VBA.UCase(sigRow(i)), Stree
Len(Street(j)) - 1

    'If its a po box then add 5 chars
    If VBA.Right(Street(j), 3) = "BOX" Then SpaceInNam

    If ActiveSheet.Shapes("chkConfirm").ControlFormat.
    ActiveSheet.Range("Street") = VBA.Mid(sigRow(i), 1
    Else
      If MsgBox("Street Address: " & VBA.Mid(sigRow(i)
vbQuestion + vbYesNo, "Confirm Details") = vbYes Then
ActiveSheet.Range("Street") = VBA.Mid(sigRow(i), 1, Sp
    End If
    'Trim the Street, Number leaving the Suburb if its
    sigRow(i) = VBA.Mid(sigRow(i), SpaceInName) + 2
    sigRow(i) = Replace(sigRow(i), VBA.Mid(sigRow(i),

    GoTo PastAddress:
    End If
    Next j
End If
Next i
PastAddress:

'Mobile
For i = 1 To UBound(sigRow)
If Len(sigRow(i)) > 0 Then
    For j = 0 To 3
    Temp = Mb(j)
        If VBA.Left(VBA.UCase(sigRow(i)), Len(Temp)) =
        If ActiveSheet.Shapes("chkConfirm").ControlFor
        ActiveSheet.Range("Mobile") = VBA.Mid(sigRow(i
        Else
          If MsgBox("Mobile: " & VBA.Mid(sigRow(i), Le
+ vbYesNo, "Confirm Details") = vbYes Then ActiveSheet
VBA.Mid(sigRow(i), Len(Temp) + 2)
        End If
    sigRow(i) = ""
    GoTo PastMobile:
    End If
    Next j
End If
Next i
PastMobile:

'Phone
For i = 1 To UBound(sigRow)
If Len(sigRow(i)) > 0 Then
    For j = 0 To 1
    Temp = Ph(j)
        If VBA.Left(VBA.UCase(sigRow(i)), Len(Temp)) =

            'TODO: Detect the intl or national extensi
from the postcode.
            If ActiveSheet.Shapes("chkConfirm").Contro
            ActiveSheet.Range("Phone") = VBA.Mid(sigRo
            Else
              If MsgBox("Phone: " & VBA.Mid(sigRow(i),
vbQuestion + vbYesNo, "Confirm Details") = vbYes Then
```

```
ActiveSheet.Range("Phone") = VBA.Mid(sigRow(i), Len(Te
            End If

        sigRow(i) = ""
        GoTo PastPhone:
        End If
    Next j
End If
Next i
PastPhone:


'Email
For i = 1 To UBound(sigRow)
    If Len(sigRow(i)) > 0 Then
        'replace with regEx search
        If InStr(1, sigRow(i), "@", vbTextCompare) And
VBA.UCase(sigRow(i)), ".CO", vbTextCompare) Then
        Dim email As String
        email = sigRow(i)
        email = Replace(VBA.UCase(email), "EMAIL:", ""
        email = Replace(VBA.UCase(email), "E-MAIL:", "
        email = Replace(VBA.UCase(email), "E:", "")
        email = Replace(VBA.UCase(Trim(email)), "E ",
        email = VBA.LCase(email)

            If ActiveSheet.Shapes("chkConfirm").Contro
            ActiveSheet.Range("Email") = email
            Else
              If MsgBox("Email: " & email, vbQuestion
Details") = vbYes Then ActiveSheet.Range("Email") = em
            End If
        sigRow(i) = ""
        Exit For
        End If
    End If
Next i

'Now the only remaining items will be the postcode, su
'there shouldn't be any numbers (eg. from PoBox,Ph,Fax
Post Code

'Join the string and filter out the Post Code
Temp = Join(sigRow, vbCrLf)
Temp = Trim(Temp)

For i = 1 To Len(Temp)

Dim postCode As String
postCode = VBA.Mid(Temp, i, 4)

'In Australia PostCodes are 4 digits
If VBA.Mid(Temp, i, 1) <> " " And IsNumeric(postCode)

    If ActiveSheet.Shapes("chkConfirm").ControlFormat.
    ActiveSheet.Range("PostCode") = postCode
    Else
      If MsgBox("Post Code: " & postCode, vbQuestion +
Details") = vbYes Then ActiveSheet.Range("PostCode") =
    End If

    'Lookup the Suburb and State based on the PostCode
the lookup
    Dim mySuburbArray As Range
    Set mySuburbArray = Sheets("PostCodes").Range("A2:

    Dim suburbs As String
    For j = 1 To mySuburbArray.Columns(1).Cells.Count
    If mySuburbArray.Cells(j, 1) = postCode Then
        'Check if the suburb is listed in the address
        If InStr(1, UCase(Temp), mySuburbArray.Cells(j
Then

        'Set the Suburb and State
        ActiveSheet.Range("Suburb") = mySuburbArray.Ce
        Stat = mySuburbArray.Cells(j, 3)
        ActiveSheet.Range("State") = Stat

        'Knowing the State - for Australia we can get
        PhExt = PhExtension(VBA.UCase(Stat))
        ActiveSheet.Range("PhExt") = PhExt

        'remove the phone extension from the number
        Dim prePhone As String
        prePhone = ActiveSheet.Range("Phone")
        prePhone = Replace(prePhone, PhExt & " ", "")
        prePhone = Replace(prePhone, "(" & PhExt & ")
        prePhone = Replace(prePhone, "(" & PhExt & ")"
        ActiveSheet.Range("Phone") = prePhone
        Exit For
        End If
    End If
    Next j
Exit For
End If
Next i

End Sub


Private Function PhExtension(ByVal State As String) As
Select Case State
Case Is = "NSW"
PhExtension = "02"
Case Is = "QLD"
PhExtension = "07"
Case Is = "VIC"
PhExtension = "03"
Case Is = "NT"
PhExtension = "04"
Case Is = "WA"
PhExtension = "05"
Case Is = "SA"
PhExtension = "07"
Case Is = "TAS"
PhExtension = "06"
End Select
End Function

Private Function Ph(ByVal Num As Integer) As String
Select Case Num
Case Is = 0
Ph = "PH"
Case Is = 1
Ph = "PHONE"
'Case Is = 2
'Ph = "P"
End Select
End Function

Private Function Mb(ByVal Num As Integer) As String
Select Case Num
```

```
Case Is = 0
Mb = "MB"
Case Is = 1
Mb = "MOB"
Case Is = 2
Mb = "CELL"
Case Is = 3
Mb = "MOBILE"
'Case Is = 4
'Mb = "M"
End Select
End Function

Private Function Fax(ByVal Num As Integer) As String
Select Case Num
Case Is = 0
Fax = "FAX"
Case Is = 1
Fax = "FACSIMILE"
'Case Is = 2
'Fax = "F"
End Select
End Function

Private Function State(ByVal Num As Integer) As String
Select Case Num
Case Is = 0
State = "NSW"
Case Is = 1
State = "QLD"
Case Is = 2
State = "VIC"
Case Is = 3
State = "NT"
Case Is = 4
State = "WA"
Case Is = 5
State = "SA"
Case Is = 6
State = "TAS"
End Select
End Function

Private Function Street(ByVal Num As Integer) As Strin
```

```vb
Select Case Num
Case Is = 0
Street = " ST"
Case Is = 1
Street = " RD"
Case Is = 2
Street = " AVE"
Case Is = 3
Street = " AV"
Case Is = 4
Street = " CRES"
Case Is = 5
Street = " LOOP"
Case Is = 6
Street = "PO BOX"
Case Is = 7
Street = " STREET"
Case Is = 8
Street = " ROAD"
Case Is = 9
Street = " AVENUE"
Case Is = 10
Street = " CRESENT"
Case Is = 11
Street = " PARADE"
Case Is = 12
Street = " PDE"
Case Is = 13
Street = " LANE"
Case Is = 14
Street = " COURT"
Case Is = 15
Street = " BLVD"
Case Is = 16
Street = "P.O. BOX"
Case Is = 17
Street = "P.O BOX"
Case Is = 18
Street = "PO BOX"
Case Is = 19
Street = "POBOX"
End Select
End Function
```

Share Follow

edited Dec 31, 2021 at 16:38

Michael come lately
9,293 ● 7 ● 66 ● 92

answered Aug 25, 2019 at 11:40

Jeremy Thompson
65.4k ● 37 ● 220 ● 338