Example: fill a form's address fields automatically

INTRODUCTION

In the previous example, we used the results returned by the Google reverse geocoding service, without going into detail.

In this section, we will see how we can get the different parts of the responses (city, street, zip code, country, etc.) The reverse geocoding service tries to guess what is the "best" address that matches the longitude and latitude, but sometimes the first guess is not the best one.

HOW TO PARSE THE GOOGLE REVERSE GEOCODING RESULTS?

What are the Google reverse geocoding results exactly?

A common question is: how to have a robust code for parsing the Google reverse geocoding, to properly get the city, street, country, etc.

Depending on your location/country and on the geolocation method used by your browser (GPS on phone, IP, WiFi, 3G, etc.), some of the data might not be available (i.e., no street). So, there is no guarantee that all candidate addresses will get the same defined properties. For example, the first result may give a defined city, but the third result may not.

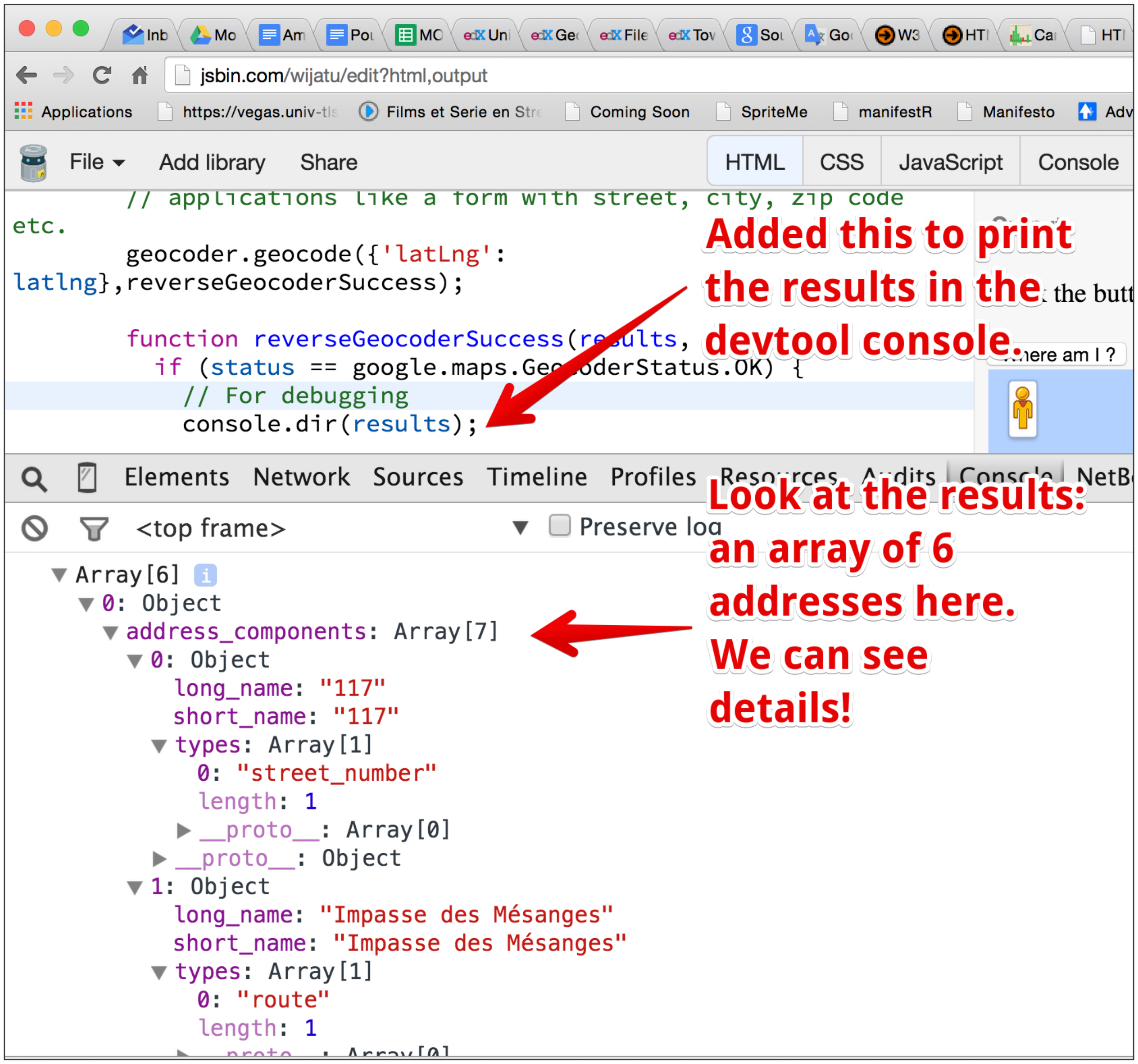
Look at this line of code from the last example from the previous page - the example that showed the address when you clicked on the button:

```
// Display address as text in the page
myAddress.innerHTML="Adress: " +results[0].formatted_address;
```

At line 2, we get the first address returned by the Google reverse geocoding service, and use the `formatted_address`property. Let's suppose that it contained the best address, formatted as a string. We chose to use it and showed it in the page by setting `myAddress.innerHTML` with its value (`myAddress` pointed to the `<p id="address"></p>` element in the page).

Let's examine the detailed results

We add a `console.dir(results)` in the code, to see a structured view of the `results` in dev. tools console:


jsbin.com/wijatu/edit?html,output
File
Add library
Share
HTML
CSS
JavaScript
Console
Elements
Network
Sources
Timeline
Profiles
Preserve log
Added this to print the results in the devtool console.
Look at the results: an array of 6 addresses here. We can see details!

```
    // applications like a form with street, city, zip code
etc.
    geocoder.geocode({'latLng':
latlng},reverseGeocoderSuccess);

    function reverseGeocoderSuccess(results,
      if (status == google.maps.GeocoderStatus.OK) {
        // For debugging
        console.dir(results);
```

```
▼ Array[6]
  ▼ 0: Object
    ▼ address_components: Array[7]
      ▼ 0: Object
          long_name: "117"
          short_name: "117"
        ▼ types: Array[1]
            0: "street_number"
            length: 1
          ▶ __proto__: Array[0]
        ▶ __proto__: Object
      ▼ 1: Object
          long_name: "Impasse des Mésanges"
          short_name: "Impasse des Mésanges"
        ▼ types: Array[1]
            0: "route"
            length: 1
```

Once we get the results, we can get the different parts:

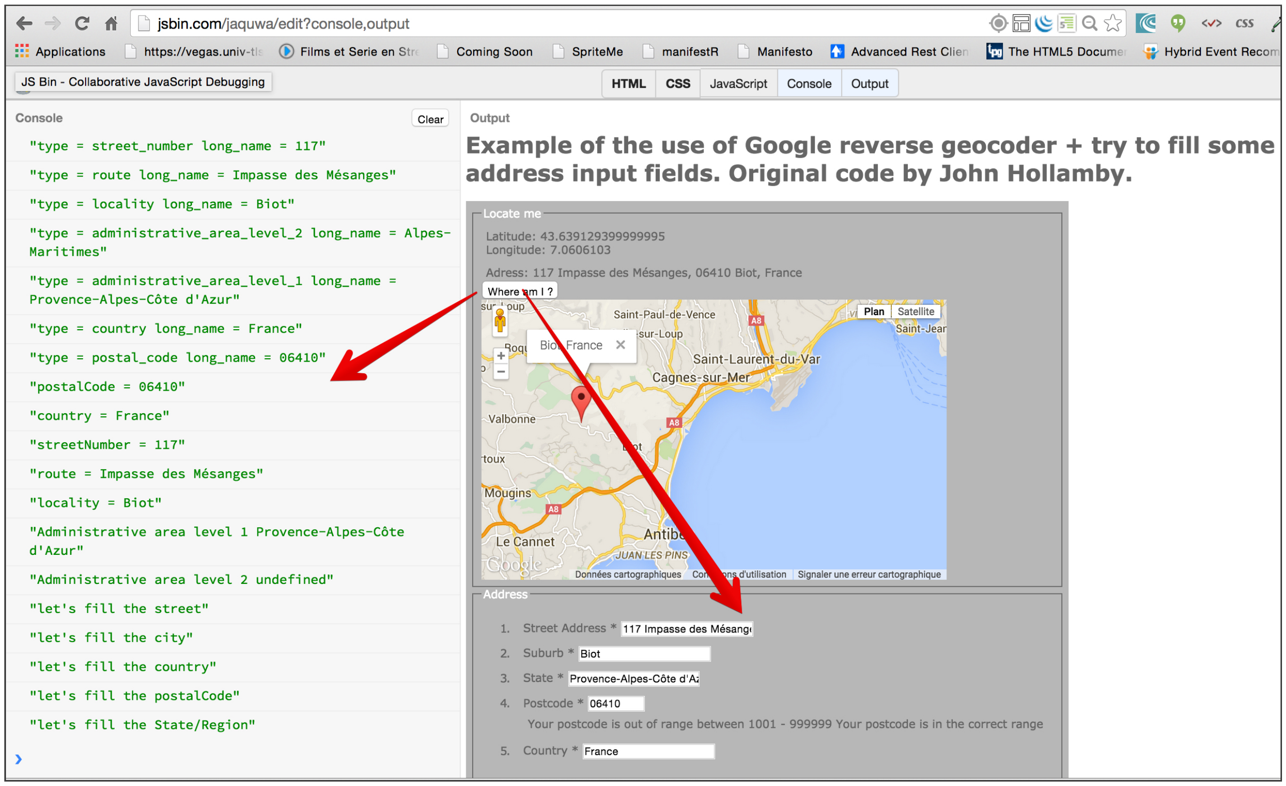
Here is an example of how we can parse such a field. Notice that each field is tested to see if it exists. The results are stored in the variables defined at line 1.

```
var country, postalCode, state, route,streetNumber, locality, areaLvl1,areaLvl2;
 function parseResult(result) {
     for(i in result){
         console.log("type = " +result[i].types[0] + " long_name = " +
                        result[i].long_name);
         if(result[i].types[0] =='postal_code')
             postalCode =result[i].long_name;
         if(result[i].types[0] =='country')
10.          country=result[i].long_name;
         if(result[i].types[0] =='street_number')
             streetNumber=result[i].long_name;
```

```
            if(result[i].types[0] =='route')
                route= result[i].long_name;
            if(result[i].types[0] =='locality')
                locality=result[i].long_name;
            if(result[i].types[0] =='state')
                state= result[i].long_name;
            if(result[i].types[0] =='administrative_area_level_2')
                arealLvl2=result[i].long_name;
            if(result[i].types[0] =='administrative_area_level_1')
                areaLvl1=result[i].long_name;
        }
        // added this for debugging in the console
    console.log("postalCode = " +postalCode);
    console.log("country = " + country);
    console.log("streetNumber = " +streetNumber);
    console.log("route = " + route);
    console.log("locality = " +locality);
    console.log("Administrative area level 1 " + areaLvl2);
    console.log("Administrative area level 2 " + areaLvl1);
  }
```

A FORM THAT AUTO FILLS THE ADDRESS INPUT FIELDS

Example at JS Bin



```
"type = street_number long_name = 117"
"type = route long_name = Impasse des Mésanges"
"type = locality long_name = Biot"
"type = administrative_area_level_2 long_name = Alpes-
Maritimes"
"type = administrative_area_level_1 long_name =
Provence-Alpes-Côte d'Azur"
"type = country long_name = France"
"type = postal_code long_name = 06410"
"postalCode = 06410"
"country = France"
"streetNumber = 117"
"route = Impasse des Mésanges"
"locality = Biot"
"Administrative area level 1 Provence-Alpes-Côte
d'Azur"
"Administrative area level 2 undefined"
"let's fill the street"
"let's fill the city"
"let's fill the country"
"let's fill the postalCode"
"let's fill the State/Region"
```

Example of the use of Google reverse geocoder + try to fill some address input fields. Original code by John Hollamby.

Latitude: 43.639129399999995
Longitude: 7.0606103

Adress: 117 Impasse des Mésanges, 06410 Biot, France

1. Street Address * 117 Impasse des Mésang
2. Suburb * Biot
3. State * Provence-Alpes-Côte d'Az
4. Postcode * 06410
   Your postcode is out of range between 1001 - 999999 Your postcode is in the correct range
5. Country * France

It's very hard to create a single code that will work in all situations and in all countries, since postal addresses are formatted differently depending on the country. A decoder that works well 99% of the time in the UK may be wrong for Australia, for instance. So, it's just a "guess system", and in real life, if you create a Web site and would like to help the user with completing a form, just fill in the country, city, postal code, and suggest the rest, propose a small icon for deleting the street input field content, etc. You could also add a drop down menu that offers not only the first guess but the second and third, etc.

Source code extract:

```
function showOnGoogleMap(latlng) {
  ...
  // Display address as text in the page
  myAddress.innerHTML="Adress: " +results[0].formatted_address;
  // Call the function that parses the results and fills
  // the input fields
 parseResult(results[0].address_components);
  ...
}
var country, postalCode, state, route,streetNumber, locality, areaLvl1,areaLvl2;

function parseResult(result) {
   for(i in result){
      // Let's print all the data we can collect from the reverse geocoder,
      // Look at the debug console to see what we get...
      console.log("type = " +result[i].types[0] + " long_name = " +
                                result[i].long_name);

      if(result[i].types[0] =='postal_code')
         postalCode =result[i].long_name;
      ...
      // fill input fields now, check if variables are undefined
      if((route != undefined) &&(streetNumber != undefined)) {
         console.log("let's fill the street");
         document.querySelector("#address1").value= streetNumber + "
" + route;
      }
      if(locality != undefined) {
         console.log("let's fill the city");
         document.querySelector("#address2").value= locality;
      }
      if(country != undefined) {
         console.log("let's fill the country");
         document.querySelector("#country").value= country;
      }
      ...
   }
}
```

```
    </script>
```

This example is rather long and we have only shown an extract of the source code. Take your time and look at the online example.