Example: fill a form's address fields automatically

INTRODUCTION

In the previous example, we used the results the Google reverse geocoding service returned, 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" surface 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 how your browser use to geolocate yourself (GPS on phone, IP, WiFi, 3G, etc.), some of the data might not be available (i.e., no street). So, it is not guaranteed that all candidate addresses will get the same defined properties. For example, the first result may give a defined city, but not the third one.

Look at this line of code from the last example from the previous page, the one that showed the surface address when you clicked on the button:
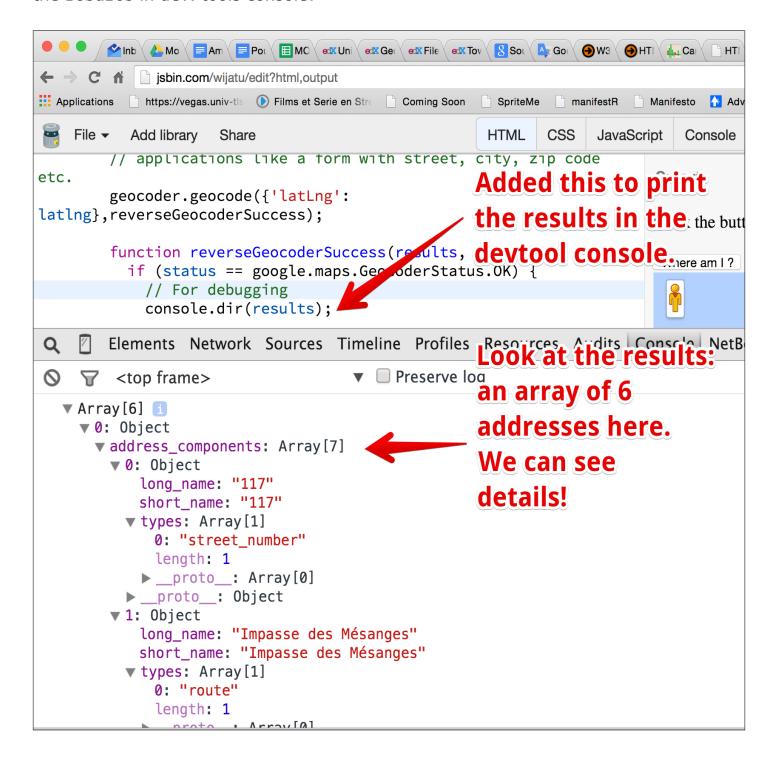
```
// Display address as text in the page
myAddress.innerHTML="Adress: " +results[0].formatted_address;
```

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 look into the detailed results

We add a `console.dir(results)` in the code, to see a structured view of the `results` in dev. tools console:
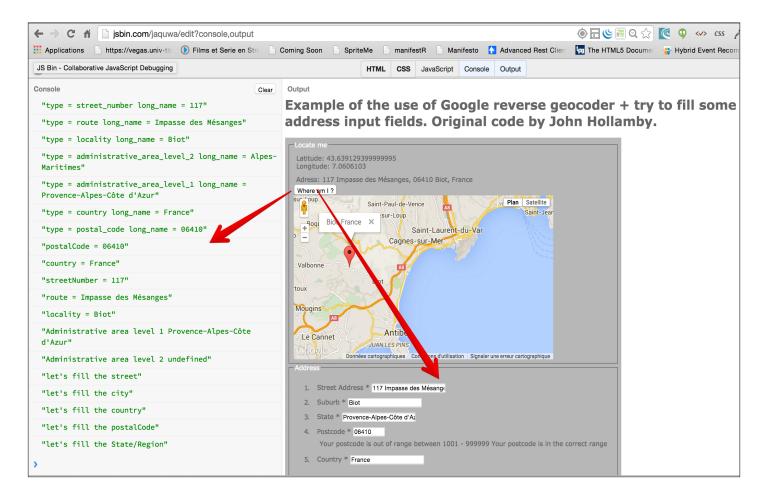
Once we get the results, here is how to get the different parts:

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

```javascript
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')
        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](#)

We forgot to say that it's very hard to have such a code to work in all situations, in all countries, as post addresses are formatted differently depending on the country. A decoder that will work 99% of the time well for UK may be wrong for Australia. So, it's just a "guess system", and in real life, if you create a Web site and would like to help filling 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. Maybe add a drop down menu for proposing not only the first guess but the second and third, etc.

Source code extract:

```javascript
function showOnGoogleMap(latlng) {
    ...
    // Display address as text in the page
    myAddress.innerHTML="Adress: " +results[0].formatted_address;
```

```javascript
      // 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;
```
11.
```javascript
    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;
```
21.
```javascript
        ...
        // 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;
        }
```
28.
```javascript
        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 only showed an extract of the source code. Take

your time and look at the online example..