Short guide to producing a Google Fusion chloropleth map - supported by interactive webpage

Robert Fry, Office for National Statistics

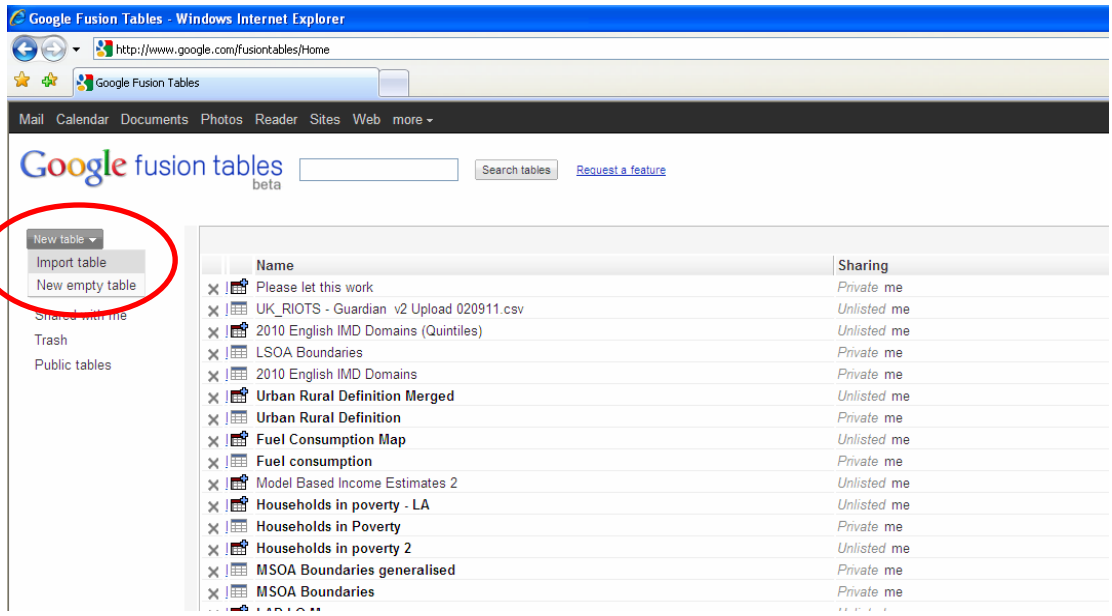
This short guide gives brief instructions on how to recreate this Google map http://www.neighbourhood.statistics.gov.uk/HTMLDocs/incomeestimates.html showing small area income estimates which was recently released in June 2011 by ONS Regional and Local Division as a prototype way of presenting information at small area level. This guide is in response to a number of queries asking how the map was made and whether it could be applied elsewhere. Hopefully this will help others.

The map has been solely developed by me, with help from examples and forums on the web. We have also consulted our geography department over licensing issues and our data visualisation unit about the colour schemes and other visualisation matters. I am not a coding expert and so much of the development has been self taught and a case of trial and error. This is also a prototype - there are things we would like to add and/or change but this will be done as time and resource permits.

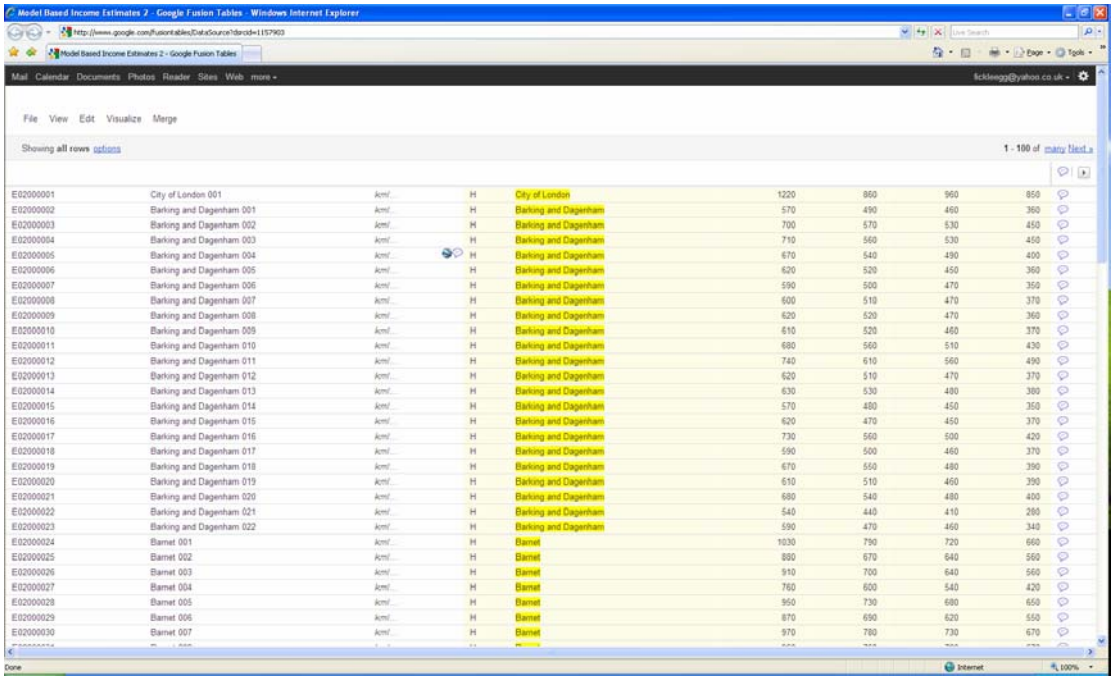
Instructions

Go to the Google Fusion home www.google.com/fusiontables/Home

Import the data you wish to map



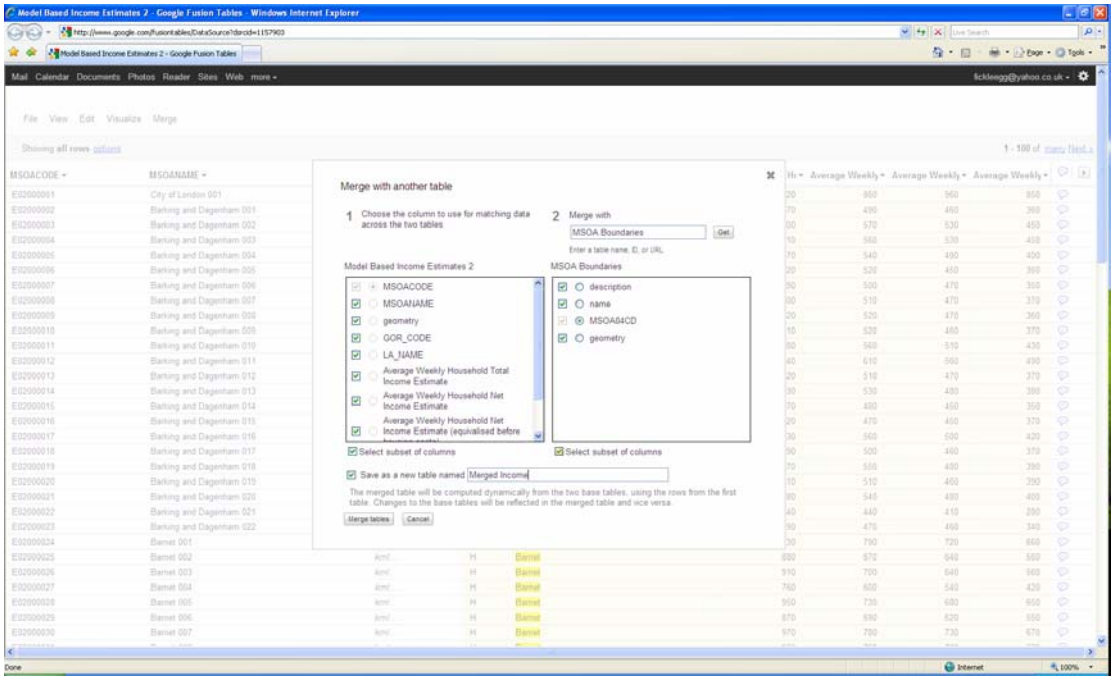
Go through the wizard steps to upload the data – you'll get a screen like this when you have uploaded the data.



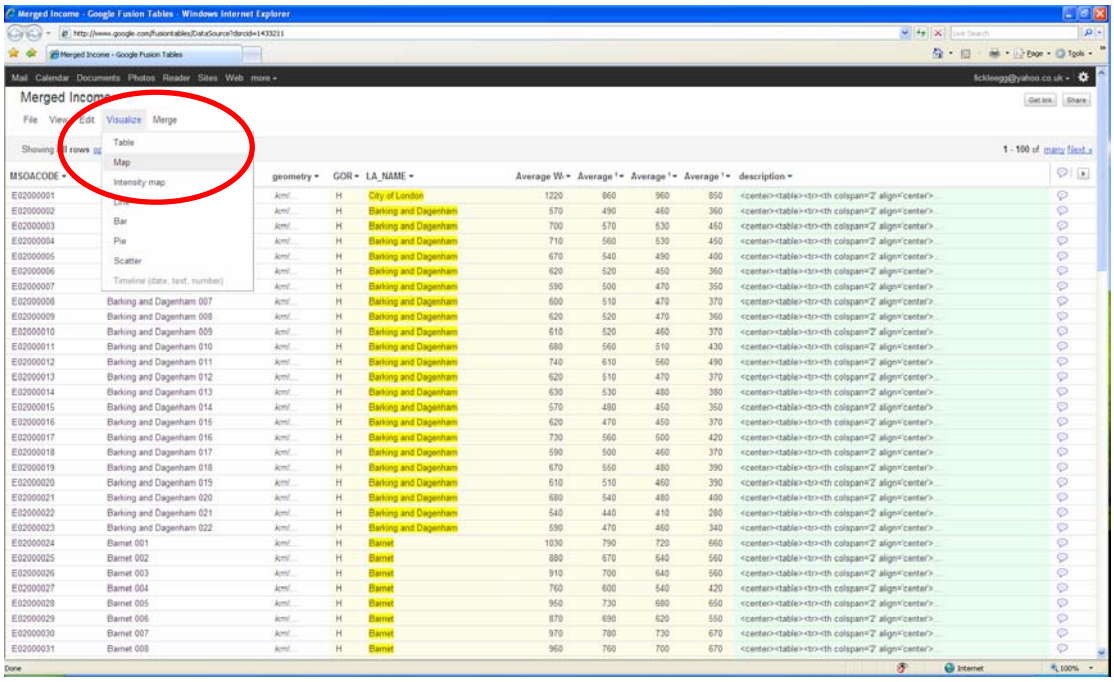
You will need to repeat this step to upload the polygon data (equivalent to the shapefile in ArcMap). For the polygon data to work in Google it needs to be in a KML format. There are various conversion tools on the web to convert a shapefile to KML.

You then need to merge the two datasets together (the data and the polygon information) based on the unique identifier on each table (eg MSOA code).

Go to the merge menu on the top bar and then follow steps shown in the wizard to create your new merged table. This involves firstly clicking on the unique identifier from each table and then selecting the columns from both tables that you wish to keep on the new table. You will need to give the new table a name.



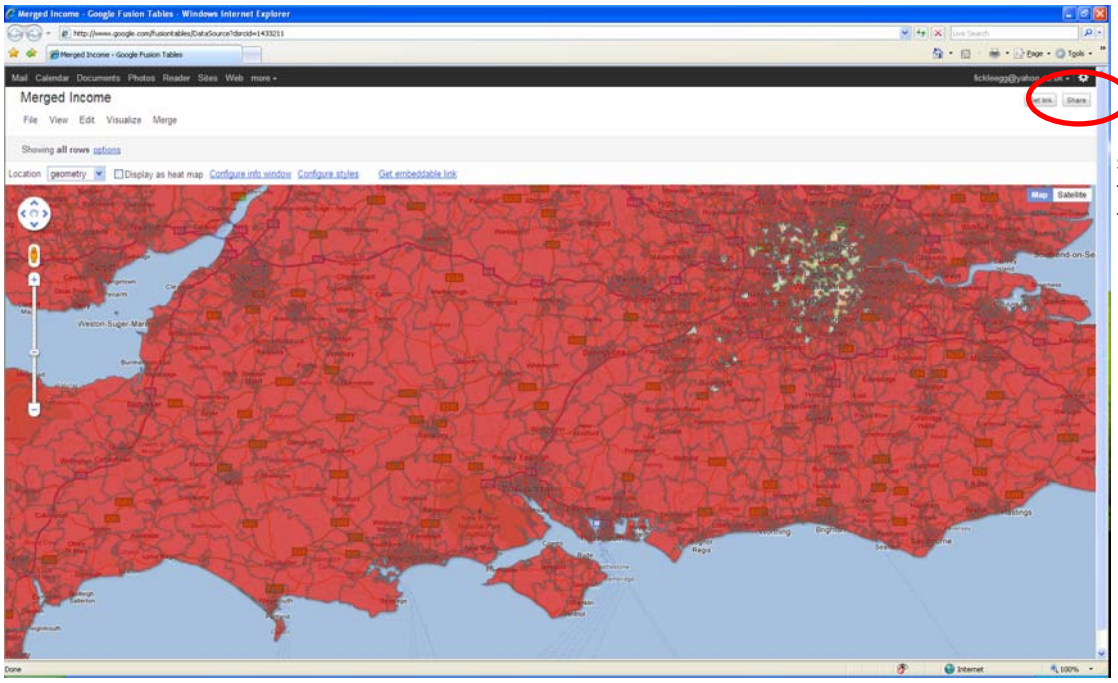
Once your new merged table has been created you can view this on a map, using the visualise menu.



Here is what you might get.

You can play around in the various menus to change how the data are displayed and coloured but using this approach is rather limiting, it's worth experimenting with though as it gives you an idea of what can be done with the mapping side.

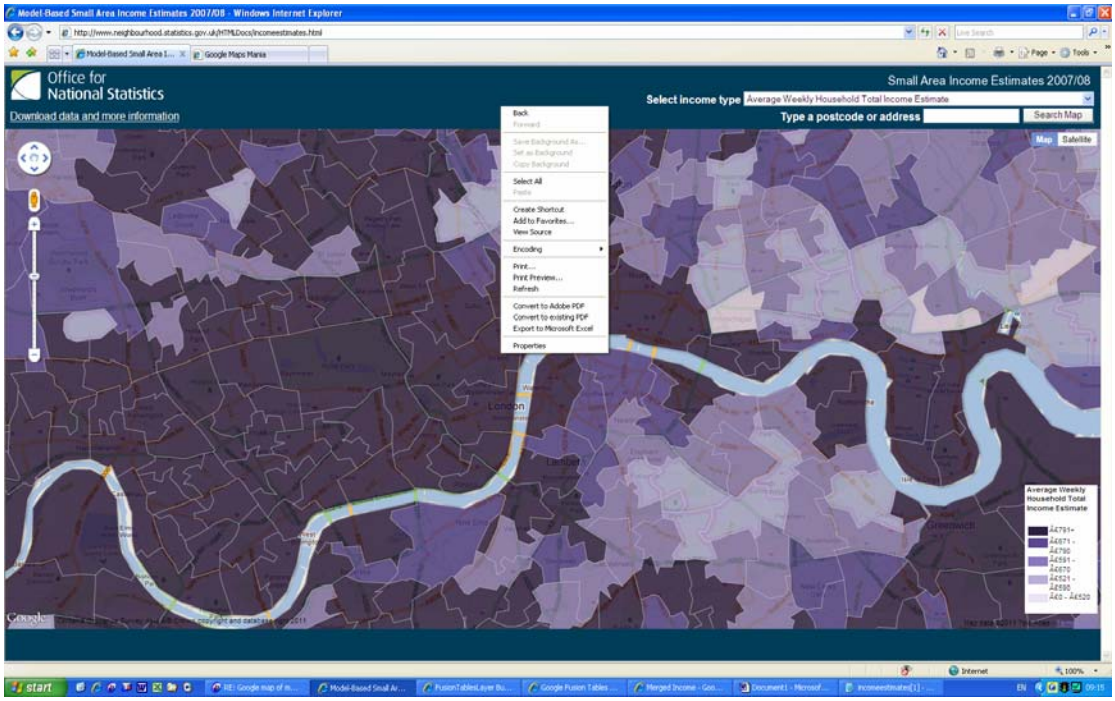
At present you are unable to create a legend and to do some of the fancier things such as automatically changing the map based on the variable you want to display (eg income type), or adding a search facility. To do this you would need to create a separate webpage and write/adapt html code to add this functionality.



Use the share menu to change the visibility

Before we get to the html code you will need to make the table/map you have created in Google Fusion available…this means changing the visibility to either 'Unlisted' or 'Public' – the wizard gives you some information on what this means.

Rather than writing the html code from scratch it would be easier to adapt code from a completed example. The following steps show you how this could be done.



Firstly acquire the code from this webpage - http://www.neighbourhood.statistics.gov.uk/HTMLDocs/incomeestimates.html

Right-click somewhere on the top of the map webpage and click view source. Save this somewhere on your PC and this becomes your base to adapt.

I'll go through the things you need to adapt starting from the top of this code.

Change the title of the page to what you would want – this is what is displayed on the top bar of your internet explorer (other browsers are available). In this example it is currently 'Model-Based Small Area Income Estimates 2007/08'

```
  </style>

  <title>Model-Based Small Area Income Estimates 2007/08</title>
  <script type="text/javascript" src="http://maps.google.com/maps/api/js?sensor=false&region=GB"></script>
  <script type="text/javascript">
```

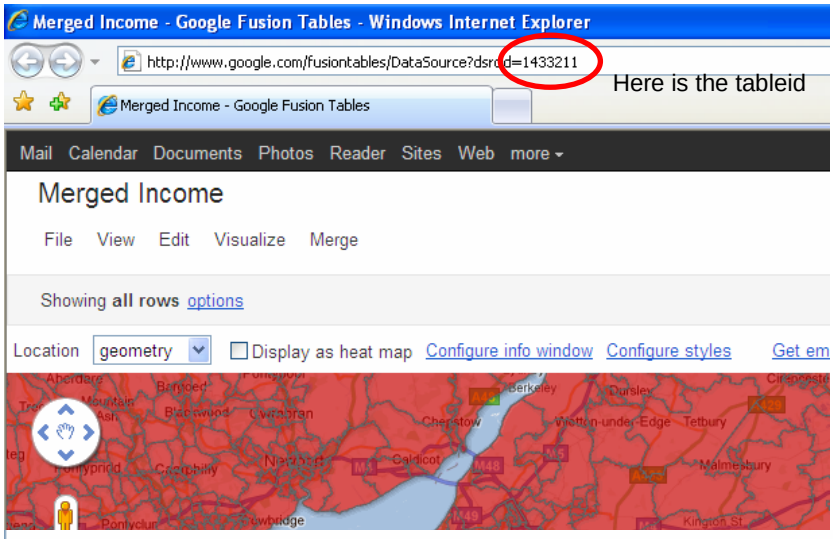
Change the items you want in the drop down menu – in this example I have the four different income measures we release – denoted by the 'divisions code'. If you need more options then copy and paste and create – I don't think there is any limit. If you need less then delete. The names of the divisions need to match the column names in your Google Fusion table exactly – this is worth bearing in mind before you upload into Google (eg Average weekly Household Total Income Estimate was a column name in my table). You also need to define the breaks in your data that you wish to display on your map. In the income example I used quintiles. The very upper and lower bounds are specified else where but for the first division you'll see I have defined ranges of:

< 520.1
520.1 <= income < 590.1
590.1 <= income < 670.1
670.1 <= income < 790.1
>=790.1

```
divisions['Average Weekly Household Total Income Estimate'] = [520.1, 590.1, 670.1, 790.1];
colors['Average Weekly Household Total Income Estimate'] = ['#E8E4F2','#B9AED8', '#8C78BE', '#5D4793', '#2A2042'];

divisions['Average Weekly Household Net Income Estimate'] = [440.1, 490.1, 540.1, 620.1];
colors['Average Weekly Household Net Income Estimate'] = ['#E8E4F2','#B9AED8', '#8C78BE', '#5D4793', '#2A2042'];

divisions['Average Weekly Household Net Income Estimate (equivalised before housing costs)'] = [400.1, 440.1, 480.1, 560.1];
colors['Average Weekly Household Net Income Estimate (equivalised before housing costs)'] = ['#E8E4F2','#B9AED8', '#8C78BE', '#5D4793', '#2A2042'];

divisions['Average Weekly Household Net Income Estimate (equivalised after housing costs)'] = [340.1, 380.1, 430.1, 490.1];
colors['Average Weekly Household Net Income Estimate (equivalised after housing costs)'] = ['#E8E4F2','#B9AED8', '#8C78BE', '#5D4793', '#2A2042'];
```

You can also change the colours should you wish using different hex values. http://colorbrewer2.org/ is a good site for choosing an accessible colour scheme.

The next step is to change the tableid – the tableid is a unique number given to your Google fusion merged table you created.




```
var tableid = 991166;
var map, layer, geocoder;
var center = new google.maps.LatLng(51.503502,-0.114069);
var zoom = 13;
```

You can also change the latitude and longitude that you want to start at. I found this tool useful for identifying the latlng of a particular point http://itouchmap.com/latlong.html

You can also change the zoom level you want to start at – the higher the number the more zoomed in you are. 1 – is the world, I think 21 is the max.

```
function initialize() {
  geocoder = new google.maps.Geocoder();
  map = new google.maps.Map(document.getElementById('map_canvas'), {
    center: center,
    zoom: zoom,
    mapTypeId: google.maps.MapTypeId.ROADMAP
  });

  layer = new google.maps.FusionTablesLayer({
    query: {
      select: "geometry",
      from: tableid,
      where: "'Average Weekly Household Total Income Estimate' > 0"
    }
  });
```

Check that the column that has your geographic data in matches what is here – I've always called it geometry but you can call it anything you like.

```
    select: "geometry",
    from: tableid,
    where: "'Average Weekly Household Total Income Estimate' > 0"
  }
```

Above I've added a condition to only show those polygons where income is greater than 0 – this was because my boundary file included Scotland and NI but the estimates only cover England and Wales. If you don't have this problem then remove this statement completely. If you have the same issue then you would need to change the variable name.

This is where you change the upper and lower bounds for the conditions. In this example I've set an upper bound of 2000 and a lower bound of 0.

```
function addStyle(column) {
  var styles = [

    {
      where: generateWhere(column, divisions[column][], 2000),
      polygonOptions: {
        fillColor: colors[column][4],
        fillOpacity: 0.65,
        strokeColor: "#cccccc",
        strokeOpacity: 0.5,
        strokeWeight: 1
      }
    },
    {
      where: generateWhere(column, divisions[column][2], divisions[column][3]),
      polygonOptions: {
        fillColor: colors[column][3],
        fillOpacity: 0.65,
        strokeColor: "#cccccc",
        strokeOpacity: 0.5,
        strokeWeight: 1
      }
    },
    {
      where: generateWhere(column, divisions[column][1], divisions[column][2]),
      polygonOptions: {
        fillColor: colors[column][2],
        fillOpacity: 0.65,
        strokeColor: "#cccccc",
        strokeOpacity: 0.5,
        strokeWeight: 1
      }
    },
    {
      where: generateWhere(column, divisions[column][0], divisions[column][1]),
      polygonOptions: {
        fillColor: colors[column][1],
        fillOpacity: 0.65,
        strokeColor: "#cccccc",
        strokeOpacity: 0.5,
        strokeWeight: 1
      }
    },
    {
      where: generateWhere(column, 0, divisions[column][0]),
      polygonOptions: {
        fillColor: colors[column][0],
        fillOpacity: 0.65,
        strokeColor: "#cccccc",
        strokeOpacity: 0.5,
        strokeWeight: 1
      }
    }
  ];
```

This line of code controls the width of the legend – this might be useful to change to make sure all your text fits on a single line etc.

```
function Legend(controlDiv, column) {
  controlDiv.style.padding = '5px';
  var controlUI = document.createElement('div');
  controlUI.style.backgroundColor = 'white';
  controlUI.style.borderStyle = 'solid';
  controlUI.style.borderWidth = '0px';
  controlUI.style.width = '110px';
  controlUI.title = 'Legend';
  controlDiv.appendChild(controlUI);
  var controlText = document.createElement('div');
  controlText.style.fontFamily = 'Arial,sans-serif';
  controlText.style.fontSize = '11px';
  controlText.style.paddingLeft = '3px';
  controlText.style.paddingRight = '3px';
  controlText.innerHTML = '<p><b>' + column + '</b></p>' +
```

The following code generates your legend – you may want to play about with what text you use – mine is based on the numbers given in the divisions, together with a little bit of text dividing the divisions.

```
'<div style="background-color: ' + colors[column][4] + '; height: 14px; width: 30px; margin: 2px; float: left;"></div>' +
' &pound' + eval(divisions[column][3]+0.9) + '+<br style="clear: both;" />' +
'<div style="background-color: ' + colors[column][3] + '; height: 14px; width: 30px; margin: 2px; float: left;"></div>' +
' &pound' + eval(divisions[column][2]+0.9) + ' - &pound' + eval(divisions[column][3]-0.1) + '<br style="clear: both;" />' +
'<div style="background-color: ' + colors[column][2] + '; height: 14px; width: 30px; margin: 2px; float: left;"></div>' +
' &pound' + eval(divisions[column][1]+0.9) + ' - &pound' + eval(divisions[column][2]-0.1) + '<br style="clear: both;" />' +
'<div style="background-color: ' + colors[column][1] + '; height: 14px; width: 30px; margin: 2px; float: left;"></div>' +
' &pound' + eval(divisions[column][0]+0.9) + ' - &pound' + eval(divisions[column][1]-0.1) + '<br style="clear: both;" />' +
'<div style="background-color: ' + colors[column][0] + '; height: 14px; width: 30px; margin: 2px; float: left;"></div>' +
' &pound0 - &pound' + eval(divisions[column][0]-0.1) + '<br style="clear: both;"/><br />';
```

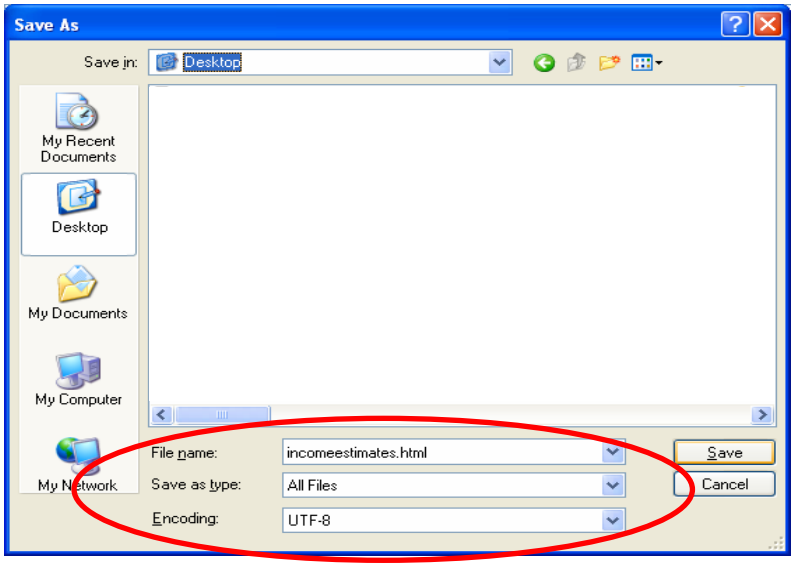
This code loads everything you have defined above and creates the webpage. You'll need to alter some of the text to adjust for your example

```
</script>
</head>
<body onload="initialize()">
<div style="position: absolute; left: 7px; top: 5px;">
<img src="./images/onsbannerlogo.jpg" alt="Logo" width="235" height="56"/>
</div>
                <div style="position:absolute; right: 10px; top:62px;">
                        <label for="address"><b>Type a postcode or address </b></label>
                        <input id="address" type="textbox" value=""/>
                        <input type="button" value="Search Map" onclick="codeAddress()"/>
                </div>
                <div style="position:absolute; right: 10px; top: 38px;">
                        <label for="selector"><b>Select income type </b></label>
                        <select onchange="addstyle(this.value);" id="selector"></select>
                </div>
                <div style="position: absolute; right: 10px; top: 5px;">
                <img src="./images/title.png" alt="Logo" width="318" height="30"/>
                </div>
                <div style="position:absolute; left: 7px; top:66px;">
                        <a href="http://www.neighbourhood.statistics.gov.uk/dissemination/Info.do?page=analysisandg
                </div>
<div id="map_canvas" style="height:84%;top:95px"></div>
</body>
</html>
```

If you're not doing this within ONS you'll need to remove the ONS logo – you could always replace it with your own. That code is circled in blue above.

You'll then need to save the file as an html file go 'File' -> Save as -> Find the folder you want to save it in.
Change the 'Save as type:' option to 'All Files' then
In the 'File name:' box save it as the name you want but add '.html'

Make sure the encoding is UTF 8 – otherwise the webpage won't display properly.



And that's it, finished, finito. Open that file up and it should work.