Roof Outlining Tool v0.2

Introduction

The roof outlining tool is split into two distinct sections, controlled by two classes: ‘RoofMapper’ and ‘RoofDrawer’. The mapping section uses Google Maps API to allow the user to locate the roof(s) they wish to outline. The drawing section uses Google Maps Static API and a canvas library to allow the user to outline the roof(s) and generate roof data. This data can then be used with the Easy-PV PV System Quote Tool API to generate a PV system quote for the drawn roof(s).

The tools are contained in the ‘roofmap-0.2.js’ file. We have also supplied a page which shows the tools working. This comprises of a html file (index.html), a javascript file (UI.js) and a CSS file (UI.css). This page is only for demonstration purposes – you will need to create and style your own hosting page.

Dependencies : Konva.js, Google Maps API, Google Maps Static API,

Class RoofMapper

The RoofMapper class is intended firstly to geolocate an address provided by the user, and secondly to display a google map centred on the geolocated address to allow the user to adjust the map to the exact location of their property.

Initialization

```
let roofMapper = new RoofMapper(container, statusCallback);
```

Upon instantiation, a map will be generated within the given container element.

Arguments:
container : An element in which the map will be contained. The map’s size will be adjusted to fill the container.
statusCallback : A function which is called when roofMapper’s status changes. It will be passed one argument which is the new status. In the demo UI.js script, we use the function ‘roofMapperStatusCallback’ as the callback, which console logs the new status, displays it in an element on the page, and provides some context-sensitive help to the user.

Using RoofMapper.setStatus()

Geolocation can be prompted using RoofMapper.setStatus(‘geolocate’, address). In response, RoofMapper will change its status to either ‘geolocate success’ or ‘geolocate fail’. These can be handled appropriately by the user within the statusCallback(status) function.

Arguments:
status: A string, as displayed in the table below. Only a status that can be set by the user is a valid argument.

address: A string, with each line of address separated by a comma. If no address is supplied, it will default to ‘’, and the geolocation will fail. Although the tool will geolocate any given address, we recommend that the first line of the address and the postcode are used, as this gives the most accurate results. You need to collect the postcode in any case as it is a required parameter for the subsequent Easy-PV API call.

Return object parameters:
success: A boolean which reflects whether or not the status was changed succesfully.
error: A string which contains error details. Only present if success = false.
status: The resulting new status.

Example:

```
let address = ‘Cambridge Industrial Estate, CB24 6AZ’;
roofMapper.setStatus(‘geolocate’, address);
// Returns {success: true, status:’geolocate’}
```

Valid values of status

Status value	Resulting event	Set by
‘initialized’	N/A.	RoofMapper
‘geolocate’	Attempts to locate the inputted address.	user
‘geolocate success’	The map zooms into the address.	RoofMapper
‘geolocate fail’	N/A.	RoofMapper

Example of a statusCallback function

```
let statusCallback = function(status){
     if (status===’geolocate success’){
          console.log(‘Address located on map’);
     }
     if (status===’geolocate fail’){
          console.log(‘Address could not be found’);
     }
}
```

Class RoofDrawer

The RoofDrawer class creates an HTML 5 canvas, imports a Google Maps image to the canvas, and allows a user to outline the roofs of the building on which they wish to install PV. It also generates a data object containing the roof outlines, which can subsequently be passed to the Easy-PV API.

Initialization

```
let map = roofMapper.map;
let roofDrawer = new RoofDrawer(map, container, statusCallback);
```

Upon instantiation, a canvas with map background will be generated within the given container element. Note that the canvas uses the element id ‘roofDrawerStage’.

Arguments:
map: A Google Maps map object centered and zoomed on the roof. This can be defined using RoofMapper as shown in the code above.
container : An element in which the drawing canvas will be contained. The canvas’ size will be adjusted to fill the container. For mobile use we recommend that the container uses as much screen as possible.
statusCallback : A function which is called when roofDrawer’s status changes. It will be passed one argument which is the new status.

Using RoofDrawer.setStatus()

Communicating with an instance of RoofDrawer is done by setting its status. In some cases the status should be set externally by the user using roofDrawer.setStatus(status). For example, to let the tool know that the user wishes to add a chimney, you would set the status to ‘startObstructionChimney’. Others are set internally within RoofDrawer. For example, once the chimney has been drawn by the user, the status will change to ‘obstructionComplete’.

In both cases, the statusCallback(status) function will be called if the status has been changed succesfully .

Arguments:
status: A string, as displayed in the the table below. Only a status that can be set by the user are valid arguments.
input: An object, only present when required by the status being set (see table below).

Return object parameters:
success: A boolean which reflects whether or not the status was changed succesfully.
error: A string which contains error details. Only present if success = false.
status: The resulting new status.

Example:

```
roofDrawer.setStatus(‘startRoofGutter’);
// Returns {success: true, status: "startRoofGutter"}
```

```
roofDrawer.setStatus(‘startObstructionChimney’);
// Returns {success: false, error: "Status "startObstructionChimney" can only
be set when the roof outline has been completed.", status: "startRoofGutter"}
```

```
roofDrawer.setStatus(‘drawPanels’,{roofID: 0, panelLayout: panelLayout,
pitch:25});
// Where panelLayout is an object returned by the Easy-PV API, located in
roofs[0].panelLayout.
// Returns {success: true, status: "drawPanels"}
```

```
roofDrawer.setStatus('undo');
// Returns {success: true, status: "initialized"}
```

Valid values of status

Status value and Input	Resulting event	Set by
‘initialized’	N/A.	RoofDrawer
‘startRoofGutter’	Used to begin drawing a roof. The next click on the drawing canvas will define the first point of the roof’s gutter. Can be set when status is ‘initialized’, ‘roofComplete’ or ‘obstructionComplete’.	user
‘drawingRoofGutter’	For desktop users, a line representing the roof’s gutter will be drawn directly between the first point and the cursor. The line will follow the cursor as it moves over the canvas. The next click on the drawing canvas will define the second point of the roof’s gutter.	RoofDrawer
‘drawingRoofVertices’	The next click on the drawing canvas will define the next roof vertex. If the first vertex is clicked, the roof outline is considered complete, and the status is changed to ‘drawnLastRoofVertex’.	RoofDrawer
‘drawnLastRoofVertex’	Roof vertices become draggable, and gutter text is added to the drawing canvas.	RoofDrawer
‘roofComplete’	Roof vertices become static.	user
‘startObstructionChimney’, ‘startObstructionVelux’, ’startObstructionOther’	Used to begin drawing an obstruction. The next click on the drawing canvas will define one corner the chosen obstruction.	user
’startObstructionVent’	The next click on the drawing canvas will define the center of the vent.	user
‘drawingObstructionChimney’, ‘drawingObstructionVelux’, ’drawingObstructionOther’	For desktop users, a rectangle representing the obstruction will be drawn between the first point and the cursor. The rectangle will follow the cursor as it moves over the canvas. The next click on the drawing canvas will	RoofDrawer

Status value and Input	Resulting event	Set by
	define the opposite corner of the obstruction.	
‘drawingObstructionVent’	For desktop users, a circle representing the obstruction will be drawn using the first point as its center, and the cursor as its outer edge. The circle will follow the cursor as it moves over the canvas. The next click on the drawing canvas will define the outer edge of the obstruction.	RoofDrawer
‘obstructionComplete’	The new obstruction becomes draggable and resizeable.	RoofDrawer
‘drawingComplete’	Indicates that the roof drawing is complete, and allows the use of roofDrawer.generateData().	user

‘drawPanels’
 	
input = {
roofIndex: Integar, the index of the roof as returned from the Easy-PV API.
panelLayout: ‘panelLayout’ of the roof as returned from the Easy-PV API.
pitch: Integar, the pitch of the roof.
}
	This status should be set after using roofDrawer.generateData() and calling the Easy-PV API. Some of the parameters returned by the API are used to draw the solar panels for a specified roof onto the canvas.	user

‘panelsComplete’	Indicates that the roof panels have been succesfully drawn on the canvas.	RoofDrawer
‘undo’	Undoes the last drawing event.	user

Using RoofDrawer.generateData()

The generateData() function can be called once the roof drawing is complete. An object is returned which contains parameters required by the Easy-PV API.

Return object parameters:
success: A boolean which reflects whether or not the output was generated successfully.
error: A string which contains error details. Only present if success = false.
postcode: The postcode of the roof(s) location.
latitude: The latitude of the roof(s) location.
longitude: The longitude of the roof(s) location.
roofs: An array containing object(s) which define the outlined roof(s). Each roof object contains its vertices and obstructions (if present) in the same format as required by the Easy-PV API.

Example:

```
RoofDrawer.generateData();

// If unsuccessful, Returns an object in the following format:
{success: false, error:'Error using RoofDrawer.generateData(): You can only
generate a roof object once the drawing is complete. Status needs to be:
"drawingComplete", Current status is: "drawingRoofVertices"'}


// If successful, Returns an object in the following format:
{success: true, postcode: ‘CB24 6AZ’, latitude:52.237247, longitude:0.158021,
 roofs:[{vertices: [[15123,19390],[22545,19298],[24493,21895],
[12617,22044]],obstructions: [["velux",641,658,15679,20504]]}]}
```

Version History

This documentation is relevant for roofmap-0.2.js.

- **Version 0.1:** Released 24/09/2020.
- **Version 0.2:** Released 19/11/2020. RoofDrawer status ‘drawPanels’ added.