Displaying an Interactive Map with Python and Flask - Part 2

we started integrating HERE maps into a Python Flask, which resulted in displaying a map. If you want to start from the very beginning or would like to revisit it, you can check it out [here](https://developer.here.com/blog/here-map-with-python-flask). Now, we are going to dig a bit deeper into it by adding features and functionalities to our map.  
  
First of all, we will be re-using the code that we built in our previous blog, so you will need to go through it once.

|  |  |
| --- | --- |
|  | <script> |
|  | // Initialize the platform object: |
|  | var platform = new H.service.Platform({ |
|  | 'apikey': 'YOUR API KEY' |
|  | }); |
|  |  |
|  | const lat = 22.719568; |
|  | const long = 75.857727; |
|  |  |
|  | // Obtain the default map types from the platform object |
|  | var maptypes = platform.createDefaultLayers(); |
|  |  |
|  | // Initialize a map: |
|  | var map = new H.Map( |
|  | document.getElementById('mapContainer'), |
|  | maptypes.raster.terrain.map, |
|  | { |
|  | zoom: 10, |
|  | center: { lat: lat, lng: long } |
|  | }); |
|  | var marker = new H.map.Marker({ lat: lat, lng: long }); |
|  |  |
|  | // Add the marker to the map: |
|  | map.addObject(marker); |
|  |  |
|  | </script> |

You can see how we used raster.terrain.map, similarly we can create different combinations from the table mentioned above. For more information on raster, you may visit the [official page](https://developer.here.com/documentation/maps/dev_guide/topics/raster.html).

Making our Map Responsive

The map we created is just an image, a static one, non-responsive to touches, taps and mouse clicks. There is no fun in a static image, what we are looking for instead, is some action, where we can do something with mouse clicks like looking around for different cities and countries. For this, we have the ‘events’ module. The best thing about the events module is that you don't need to write different code for different browsers on different platforms, it handles all the complexity for you. All you need to do is to use it and not worry about anything else.

 How are we going to use it? By loading a module called ‘mapsjs-core.js’ in the <head> of HTML file -

|  |  |
| --- | --- |
|  | <script src="https://js.api.here.com/v3/3.1/mapsjs-core.js" type="text/javascript" charset="utf-8"></script> |

[**view raw**](https://gist.github.com/vidhanbhonsle/9ea9d30a66098e98eaf960f3d943ffd5/raw/16f48c0e999ef8e414e154d47c6b0a1b1c59b924/scripts.html)[**scripts.html**](https://gist.github.com/vidhanbhonsle/9ea9d30a66098e98eaf960f3d943ffd5#file-scripts-html) hosted with ❤ by [**GitHub**](https://github.com/)

Next, with the help of Map object, instantiate the MapEvents class and lastly, add an events listener to the map. This is how it will look like -

|  |  |
| --- | --- |
|  | // Enable the event system on the map instance: |
|  | var mapEvents = new H.mapevents.MapEvents(map); |
|  |  |
|  | // Add event listener: |
|  | map.addEventListener('tap', function(evt) { |
|  | // Log 'tap' and 'mouse' events: |
|  | console.log(evt.type, evt.currentPointer.type); |
|  | }); |

To see it working, after running the python code and opening the address http://127.0.0.1:5000, you will need to open Console on your brows

So basically you can change the events as per your need and requirement. The Events module gives you the following events -

* Pointer - pointerup, pointerdown, pointermove, pointerenter, pointerleave, pointercancel
* Drag - dragstart, drag, dragend
* Tap - tap, dbltap
* Long Press - longpress

We can also have a full interaction like pan, zoom and pinch-to-zoom by adding only one line of code -

|  |  |
| --- | --- |
|  | var behavior = new H.mapevents.Behavior(mapEvents); |

[**view raw**](https://gist.github.com/vidhanbhonsle/b03a5791d4fbddb5d81d5b68db709507/raw/c7086cba22f40b3f9f2260808400e18e95128c78/part_3.html)[**part\_3.html**](https://gist.github.com/vidhanbhonsle/b03a5791d4fbddb5d81d5b68db709507#file-part_3-html) hosted with ❤ by [**GitHub**](https://github.com/)

Adding controls and UI components to the Map

HERE provides ready-made map controls through the UI module named ‘mapsjs-ui.js’. All you have to do is add following in the in the <head> of HTML file of the existing code -

|  |  |
| --- | --- |
|  | <script src="https://js.api.here.com/v3/3.1/mapsjs-ui.js" type="text/javascript" charset="utf-8"></script> |
|  | <link rel="stylesheet" type="text/css"href="https://js.api.here.com/v3/3.1/mapsjs-ui.css" /> |

[**view raw**](https://gist.github.com/vidhanbhonsle/6921b8ece7db1154392d1a906b09b884/raw/9f3e0660a9ff2347dc9146b4eb3ee569d828a53c/part_4.html)[**part\_4.html**](https://gist.github.com/vidhanbhonsle/6921b8ece7db1154392d1a906b09b884#file-part_4-html) hosted with ❤ by [**GitHub**](https://github.com/)

Now, we have added a CSS file link too, because we want our map to look appealing.

By adding just one line of code, we get a default UI and apart from zoom-in and zoom-out feature, we get a collection of default map layers.

|  |  |
| --- | --- |
|  | // Create the default UI: |
|  | var ui = H.ui.UI.createDefault(map, maptypes); |

[**view raw**](https://gist.github.com/vidhanbhonsle/b01cd43d40563300687d0228f556efee/raw/d381083a5c93164ca466ef07ee2a4b57303010e7/part_5.html)[**part\_5.html**](https://gist.github.com/vidhanbhonsle/b01cd43d40563300687d0228f556efee#file-part_5-html) hosted with ❤ by [**GitHub**](https://github.com/)

If required, we can also add an info bubble that allows us to show a bubble at a particular position containing HTML content on the map.

|  |  |
| --- | --- |
|  | // Create an info bubble object at a specific geographic location: |
|  | var bubble = new H.ui.InfoBubble({ lng: 76.0534, lat: 22.9676 }, { |
|  | content: '<b>Dewas</b>' |
|  | }); |
|  |  |
|  | // Add info bubble to the UI: |
|  | ui.addBubble(bubble); |

[**view raw**](https://gist.github.com/vidhanbhonsle/eca11136454b22fc0015815a3af9a235/raw/2354bc4abeeceaa9a3546fc9ca6dcd28ac311ab1/part_6.html)[**part\_6.html**](https://gist.github.com/vidhanbhonsle/eca11136454b22fc0015815a3af9a235#file-part_6-html) hosted with ❤ by [**GitHub**](https://github.com/)

You can explore the UI components further [here](https://developer.here.com/documentation/maps/dev_guide/topics/map-controls-ui.html).

The complete code with all the changes will look like this  -

|  |  |
| --- | --- |
|  | <html> |
|  | <head> |
|  | <meta name="viewport" content="initial-scale=1.0, width=device-width" /> |
|  | <script src="https://js.api.here.com/v3/3.1/mapsjs-core.js"type="text/javascript" charset="utf-8"></script> |
|  | <script src="https://js.api.here.com/v3/3.1/mapsjs-service.js"type="text/javascript" charset="utf-8"></script> |
|  | <script src="https://js.api.here.com/v3/3.1/mapsjs-mapevents.js" type="text/javascript" charset="utf-8"></script> |
|  | <script src="https://js.api.here.com/v3/3.1/mapsjs-ui.js" type="text/javascript" charset="utf-8"></script> |
|  | <link rel="stylesheet" type="text/css"href="https://js.api.here.com/v3/3.1/mapsjs-ui.css" /> |
|  | </head> |
|  |  |
|  | <body style='margin: 0'> |
|  | <div style="width: 100%; height: 100%" id="mapContainer"></div> |
|  |  |
|  | <script> |
|  | // Initialize the platform object: |
|  | var platform = new H.service.Platform({ |
|  | 'apikey': 'YOUR API KEY' |
|  | }); |
|  |  |
|  | const lat = 22.719568; |
|  | const long = 75.857727; |
|  |  |
|  | // Obtain the default map types from the platform object |
|  | var maptypes = platform.createDefaultLayers(); |
|  |  |
|  | // Initialize a map: |
|  | var map = new H.Map( |
|  | document.getElementById('mapContainer'), |
|  | maptypes.raster.terrain.map, |
|  | { |
|  | zoom: 10, |
|  | center: { lat: lat, lng: long } |
|  | }); |
|  |  |
|  | // Enable the event system on the map instance: |
|  | var mapEvents = new H.mapevents.MapEvents(map); |
|  |  |
|  | // Add event listener: |
|  | map.addEventListener('tap', function(evt) { |
|  | // Log 'tap' and 'mouse' events: |
|  | console.log(evt.type, evt.currentPointer.type); |
|  | }); |
|  |  |
|  | // Instantiate the default behavior, providing the mapEvents object: |
|  | var behavior = new H.mapevents.Behavior(mapEvents); |
|  |  |
|  | //window.addEventListener('resize',()=> get.ViewProt().resize()) |
|  | var marker = new H.map.Marker({ lat: lat, lng: long }); |
|  |  |
|  | // Add the marker to the map: |
|  | map.addObject(marker); |
|  |  |
|  | // Create the default UI: |
|  | var ui = H.ui.UI.createDefault(map, maptypes); |
|  |  |
|  | // Create an info bubble object at a specific geographic location: |
|  | var bubble = new H.ui.InfoBubble({ lng: 76.0534, lat: 22.9676 }, { |
|  | content: '<b>Dewas</b>' |
|  | }); |
|  |  |
|  | // Add info bubble to the UI: |
|  | ui.addBubble(bubble); |
|  |  |
|  | </script> |
|  | </body> |
|  | </html> |

[**view raw**](https://gist.github.com/vidhanbhonsle/1980250e6307f06079558324ff2dcee9/raw/ad464ab45f8968c684eada8cfdae02539e40261b/map.html)[**map.html**](https://gist.github.com/vidhanbhonsle/1980250e6307f06079558324ff2dcee9#file-map-html) hosted with ❤ by [**GitHub**](https://github.com/)

This concludes our two parts tutorial on creating an interactive map with Python Flask.