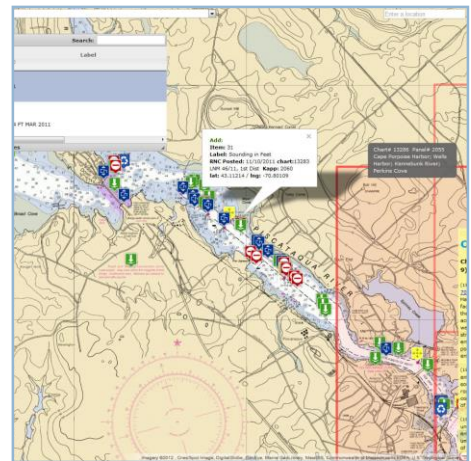


VERSION 1.0  
JUNE 11, 2014

ChUM™



## Chart Update Mashup

AUTHOR: BRIANA SULLIVAN

CENTER FOR COASTAL AND OCEAN MAPPING/JOINT HYDROGRAPHIC CENTER  
UNIVERSITY OF NEW HAMPSHIRE, DURHAM



# CHART UPDATE MASHUP

## INTRODUCTION

The Chart Update Mash up (ChUM™) consists of various APIs/Technologies that when brought together allow for intuitive georeferenced visualization of the Critical Chart Updates published by the Office of Coast Survey (OCS). Each component of ChUM™ will be described in this paper with an explanation of how each piece works together to form ChUM™. Figure 1 shows the basic building blocks that make up ChUM™. At the base of the structure (in blue) is the data that OCS disseminates to the public via its website/web-services: Raster Nautical Charts, the CRIT data (critical corrections to the chart), and the Coast Pilot. There are three 3<sup>rd</sup> party API's (Application Programming Interface)/plug-ins that are used: the Google Maps API v3, the DataTables plug-in, and the NauticalChartsAPI (NCAPI). The NCAPI is an API created by the University of California, San Diego's Coastal Observing Research and Development Center (CORDC) and it deals directly with fetching and preparing the Raster Nautical charts for use in Google Maps. The DataTables is a plug-in for the jQuery JavaScript library and it is populated with data from the OCS CRIT data. The NauticalChartsAPI UI (User Interface), NCAPIUI, was created by combining the NCAPI, DataTables, and the Coast Pilot data into one API that contained widgets for an intuitive interface that is similar in look and function as the Google Maps widgets. Google Map API is used for its geo-referenced environment and for its familiar controls that are built-in: namely, the panning, zooming, markers/info boxes and various background-overlay options.

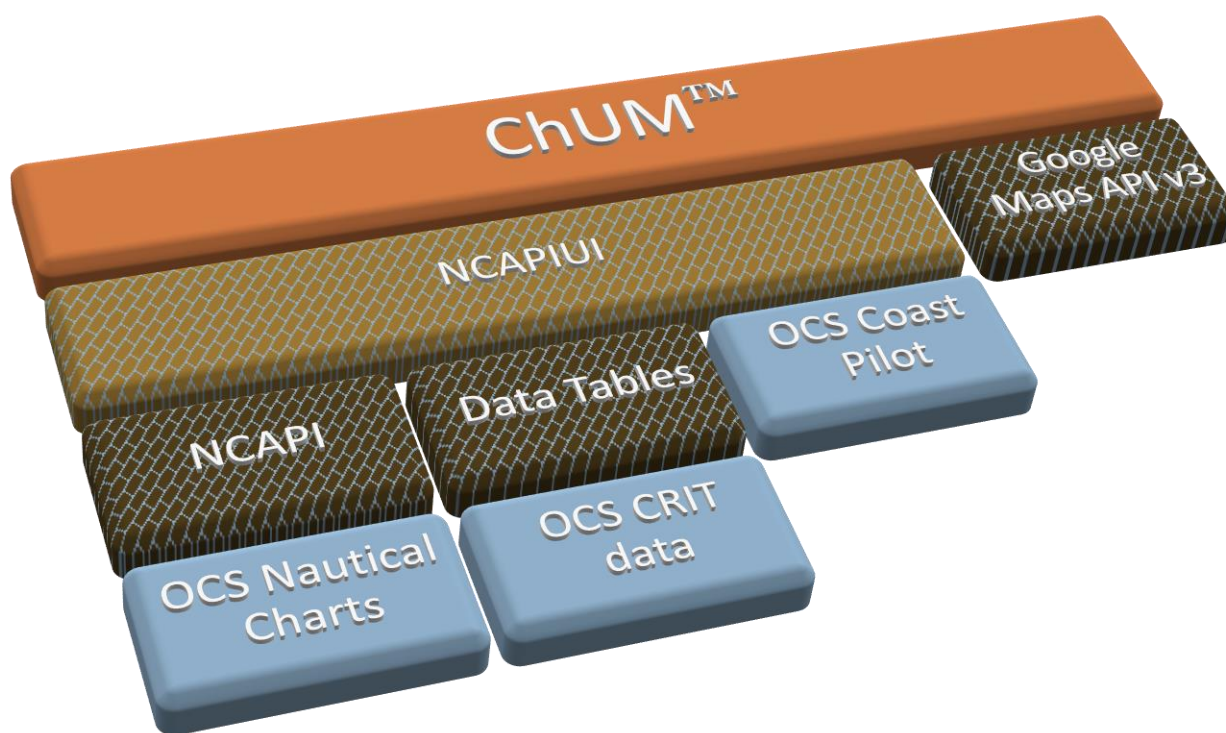


FIGURE 1. BUILDING BLOCKS USED TO CREATE CHUM™.



## OCS NAUTICAL CHARTS/CHART CATALOG

Nautical charts are necessary for the chart updates to have a context. OCS posts an XML chart catalog of the Raster Nautical Charts (RNC) at: [http://www.charts.noaa.gov/RNCs/RNCProdCat\\_19115.xml](http://www.charts.noaa.gov/RNCs/RNCProdCat_19115.xml). The chart catalog contains the metadata for each nautical chart NOAA produces. For example, chart number 11006 starts off like this:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes" ?>
<extent>
  <EX_Extent>
    <description>
      <gco:CharacterString>
        panel type: Main Panel; number: 44; title: KEY WEST TO THE MISSISSIPPI RIVER; file name: 11006_1.KAP; scale: 875000
      </gco:CharacterString>
    </description>
    <geographicElement>
      <EX_BoundingPolygon>
        <polygon>
          <gml:Polygon gml:id="US11006_P44">
            <gml:exterior>
              <gml:LinearRing>
                <gml:pos>24.16667 -90.08334</gml:pos>
                <gml:pos>30.58333 -90.08334</gml:pos>
                <gml:pos>30.58333 -88.2</gml:pos>
                <gml:pos>30.68333 -88.2</gml:pos>
                <gml:pos>30.68333 -87.9</gml:pos>
                <gml:pos>30.58333 -87.9</gml:pos>
                <gml:pos>30.58333 -81</gml:pos>
                <gml:pos>24.16667 -81</gml:pos>
              </gml:LinearRing>
            </gml:exterior>
          </gml:Polygon>
        </polygon>
      </geographicElement>
    </EX_BoundingPolygon>
  </description>
</EX_Extent>
</extent>
```

FIGURE 2. NOAA XML CHART CATALOG EXCERPT.

It is possible also to show the metadata from an individual chart, if the desired chart number is known; the URL would then be: [http://www.charts.noaa.gov/RNCs/11006\\_19115.xml](http://www.charts.noaa.gov/RNCs/11006_19115.xml) (for example). To download the actual chart the URL would be: <http://www.charts.noaa.gov/RNCs/11006.zip>, replacing the example chart number with the desired chart number. The downloaded zip file contains a directory called “BSB\_ROOT” and within that, a directory named after the chart number queried. The zipped files consist of a .BSB file and a .KAP file along with a user agreement text file.

## NAUTICAL CHARTS API

When this project began OCS allowed public access to their nautical charts in BSB format only. Since Sept 1, 2011, they released a seamless chart server which may be addressed in a future version of ChUM™. Google Maps relies on a tiling system (Figure 5) to serve up the images/overlays requested, depending upon the georeferenced bounding box of the viewing area on the map. In order to not have to “recreate the wheel” a search was done to see if anyone had done work with nautical charts set up to work with the Google Maps tiling system. The Nautical Charts API - NCAPI (Coastal Observing Research and Development Center, 2014) from the University of California, San Diego (UCSD) came up in the search results.

Paul Reuter of UCSD began development of a chart tile server in July 2008. Each night, the chart server acquires the XML NOAA RNC product catalog (Figure 2) and checks for changes. Any charts that have been modified are subsequently downloaded, tiled, and then added to a database. Static JSON data (a lightweight data-interchange format) is then generated and pushed to the UCSD web servers. The URL for the JSON data is: <http://cordc.ucsd.edu/js/RNC/rnc.php?callback=?>, and chart 11006 looks like this:

```
{
  "chart_id": "11006",
  "source_edition": "34",
  "raster_edition": "1",
  "ntm_edition": "18",
  "source_date": "2013-08-01",
  "ntm_date": "2014-04-05",
  "title": "Gulf Coast - Key West to Mississippi River",
  "panels": [
    {
      "panel_no": "44",
      "file_name": "11006_1.KAP",
      "scale": "875000",
      "max_zoom": "12",
      "encoded_polygon": "u'orCzkydPcgdfi@??(ynl_pR??_ry@-oR??_tbi@bgdfi@??zamv@",
      "encoded_levels": "PPLLLLLPPP"
    }
  ]
}
```

FIGURE 3. JSON DATA FOR NAUTICAL CHART METADATA RETURNED FROM COORD SERVERS.



Eventually, an API (NCAPI) was built around the chart server so UCSD could use it on multiple domains.

Upon initialization of the NauticalChartAPI object, `m_initializeRNCs()` is called. This function loads and parses the remote JSON chart catalog metadata using `jQuery.getJSON()`; the first argument being the url for the JSON data: <http://cordc.ucsd.edu/js/RNC/rnc.php?callback=?>, the data sent in the second argument is generated from each chart/panel that is set up from the chart catalog data that is parsed.

## COMPONENTS OF THE NAUTICAL CHARTS API

There are four classes in the Nautical Charts API, they are laid out with their local variables in Figure 4. (The arrow from a variable to the next class, shows that the variable is an array of objects of that class type.) Tables 1 – 4 show a listing of all the methods within each class, for easy reference.

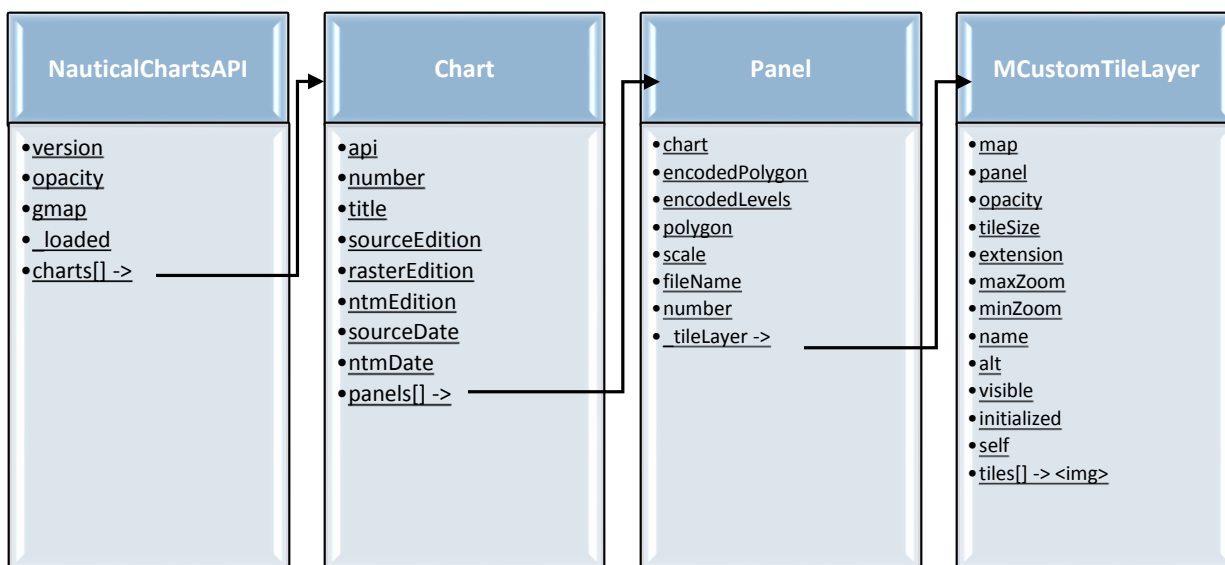


FIGURE 4. CLASSES WITHIN THE NCAPI – SHOWING VARIABLES WITHIN THE CLASS

TABLE 1. NAUTICAL CHARTS API – NAUTICALCHARTSAPI CLASS METHODS

NauticalChartsAPI Methods		
setMap()	getChartsByLatLng()	getOpacity()
isLoaded()	getPanelsByLatLng()	setOpacity()
getChartByNumber()	getChartsByBounds()	m_initializeRNCs()
getPanelByFileName()	getPanelsByBounds()	m_appendScript()
getPanelByFileNumber()	getChartsByBoundsNESW()	m_tell()
getCharts()	getPanelsByBoundsNESW()	m_wait()

TABLE 2. NAUTICAL CHARTS API - CHART CLASS METHODS

Chart Methods		
getAPI()	getSourceDate()	getPanels()
getNumber()	getNTMDate()	getPanelByFileName()



getTitle()	getBounds()	getPanelByNumber()
getSourceEdition()	hasLatLng()	getPanelsByLatLng()
getRasterEdition()	overlapsBounds()	getPanelsByBounds()
getNTMEdition()	overlapsBoundsNESW()	getPanelsByBoundsNESW()

**TABLE 3. NAUTICAL CHARTS API - PANEL CLASS METHODS**

Panel Methods		
getAPI()	getPolygon()	hasLatLng()
getChart()	getScale()	getBounds()
getEncodedPolygon()	getFileName()	overlapsBounds()
getEncodedLevels()	matchesFileName()	overlapsBoundsNESW()
getDecodedLevels()	getNumber()	getTileLayer()

**TABLE 4. NAUTICAL CHARTS API - MCustomTILELAYER CLASS METHODS**

MCustomTileLayer Methods – 2010 Marcel Montagna – <a href="http://maps.forum.nu">http://maps.forum.nu</a>		
getTile()	show()	getFileName()
initialize()	getTileUrl()	setOpacity()
hide()	getPanel()	setObjectOpacity()

## GOOGLE MAPS JAVASCRIPT API V3

Google Maps is the foundation for the NCAPI and ChUM™, it is what makes the ChUM™ application a “Mash-up”. Google Maps was the obvious choice for visualizing the geographic data that makes up the Critical Chart Corrections. The Google Maps interface is widely known, many of the needed features (zooming, georeferenced markers, ability to overlay charts, etc.) are built into the API already. The following is a description (Google Maps Developer, 2014) of how images are overlaid on the Google Map using the tiling system:

“The Google Maps API could not possibly load all map imagery at the higher zoom levels that are most useful; instead, the Maps API breaks up imagery at each zoom level into a set of map tiles, which are logically arranged in an order which the application understands. When a map scrolls to a new location, or to a new zoom level, the Maps API determines which tiles are needed using pixel coordinates, and translates those values into a set of tiles to retrieve. These tile coordinates are assigned using a scheme which makes it logically easy to determine which tile contains the imagery for any given point.

Tiles in Google Maps are numbered from the same origin as that for pixels. For Google's implementation of the Mercator projection, the origin tile is always at the northwest corner of the map, with x values increasing from west to east and y values increasing from north to



south. Tiles are indexed using x,y coordinates from that origin. For example, at zoom level 2, when the earth is divided up into 16 tiles, each tile can be referenced by a unique x,y pair:

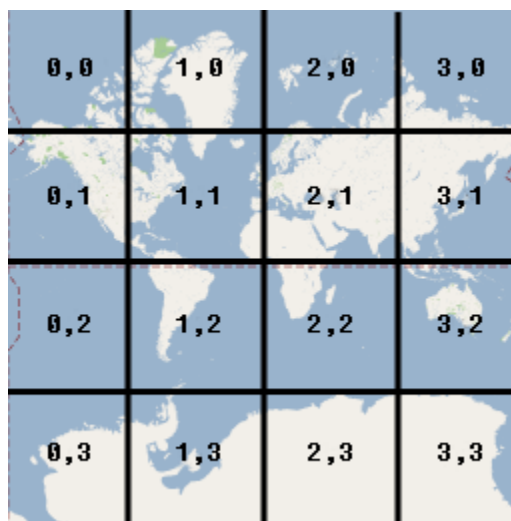


FIGURE 5. EXAMPLE OF THE GOOGLE MAPS TILING LAYOUT.

Note that by dividing the pixel coordinates by the tile size and taking the integer parts of the result, you produce as a by-product the tile coordinate at the current zoom level.”

The chart overlays from the NCAPI are using this tiling system from Google Maps.

In 2009, Google released version 3 of its JavaScript Maps API. The blog announcing its release explains why the upgrade was necessary, “The primary motivation behind this new version was speed, especially for rendering maps on mobile browsers” (Google Geo Developers, 2009). NCAPI was upgraded on 12/20/11 to use this new version of Google Maps. UCSD used this new code to upgrade their servers as well. This upgrade sped up the online performance as well as laid the foundation for a mobile version of ChUM™.

## CRITICAL CHART UPDATES FROM OCS

The NCAPI brought the nautical chart into a geo-referenced environment, laying the foundation for displaying the Local Notice to Mariner (LNM) Data. The LNM contains a subsection of data that is considered “critical” to updating nautical charts. These Critical Chart Updates (from the OCS CRIT database) have lat/lng positions attached to them. Using the Google Maps API makes it relatively easy to place geo-referenced objects as markers on the map. The most important part to generating markers in a Google Map is to have latitude and a longitude position, which can be extracted from the text output of the OCS server query:

[http://ocsdata.ncd.noaa.gov/ntm/Listing\\_Text.aspx?Chart=13282&DateSince=.](http://ocsdata.ncd.noaa.gov/ntm/Listing_Text.aspx?Chart=13282&DateSince=)

This query is called each time a new chart is selected (or when the LNM date changes, the “DateSince” query). The text returned from this query is then parsed with PHP (a server-side scripting language) and encoded as JSON data (Figure 6).





```
{
  - aaData: [
    - {
      0: "Change",
      1: "Overhead Power Cable Label to:",
      2: "OVHD PWR CAB AUTH CL 32 FT",
      3: "43.16333",
      4: "-70.85712",
      5: "LNM 25/13, 1st Dist",
      6: "2060",
      7: "6/17/2013 ",
      DT_RowId: 2
    },
    - {
      0: "Relocate",
      1: "Piscataqua River Buoy 8",
      2: "R N "8"",
      3: "43.1065",
      4: "-70.79089",
      5: "LNM 09/13, 1st Dist",
      6: "2060",
      7: "3/20/2013 ",
      DT_RowId: 3
    },
    - {
      0: "Relocate",
      1: "Piscataqua River Lighted Buoy 7",
      2: "G "7" Fl G 4s",
      3: "43.10482",
      4: "-70.79066",
      5: "LNM 09/13, 1st Dist",
      6: "2060",
      7: "3/20/2013 ",
      DT_RowId: 4
    }
  ]
}
```

FIGURE 6. LNM JSON DATA RETURNED FROM QUERY

The JSON data is what populates the Data Table, and the content of the Data Table is what drives the creation and display of the markers on the Google Map.

## DATA TABLES

DataTables, a plug-in for the jQuery JavaScript library, is a “highly flexible tool, based upon the foundations of progressive enhancement, and will add advanced interaction controls to any HTML table” (SpryMedia Ltd ©, 2014). DataTables generates all the markers representing each chart correction (Figure 7). The table also displays the total count of items in the table as well as the filtered count for easy reference - helping with data quality assurance.



Search: <input type="text"/>							
Action	Item Name	Label	Lat	Lng	Pub	Kapp	RNC
Change	Overhead Power Cable Label to:	OVHD PWR CAB AUTH CL 32 FT	43.16333	-70.85712	LNK 25/13, 1st Dist	2060	6/17/2013
Relocate	Piscataqua River Buoy 8	R N "8"	43.1065	-70.79089	LNK 09/13, 1st Dist	2060	3/20/2013
Relocate	Piscataqua River Lighted Buoy 7	G "7" Fl G 4s	43.10482	-70.79066	LNK 09/13, 1st Dist	2060	3/20/2013

Showing 1 to 3 of 3 entries

FIGURE 7. LNK JSON DATA SHOWN IN THE DATATABLE.

Two files are needed in the web-site to use Data Tables: the CSS styling and the DataTables JavaScript file which are located on the DataTables CDN site: ( SpryMedia Ltd ©, 2014). To make a data table an empty HTML table element must be created in the body of the document. Using JQuery, the data table is created by accessing the HTML table element and referencing the dataTable constructor with any desired attributes. An example of a simple initialization is:

```
$( '#myTable' ).dataTable();
```

To add the LNK JSON data the “bProcessing” attribute needs to be true, and the “sAjaxSource” attribute needs to have the location of the script that will return the JSON data, ex:

```
“scripts/server_processing.php?chart=13286&dateSince=”
```

Each time a new chart is selected or a new date has been entered the Data Table calls

```
.fnReloadAjax(“scripts/server_processing.php?chart=13286&dateSince=”,sNewSource, fnCall).
```

**Note:** ChUM™ uses DataTables version 1.9.0, fnReloadAjax is deprecated as of DataTables 1.10, which provides ajax.url() and ajax.reload() to give the same functionality.

## NAUTICAL CHARTS APIUI

The Nautical Charts APIUI (NCAPIUI) was created to put all of the preceding elements together with a user interface, enabling programmers/scientists who wish to integrate nautical charts into their own project to do so with a few lines of code. The NCAPIUI allows for quick and customized setup of the charts and the desired chart controls. Using JavaScript and the jQuery library, various features were added on top of the NCAPI that made it possible to: display the chart catalog, select/view specific charts (in a drop-down box), filter the charts depending on the viewport, see the latitude and longitude values for the cursor position, set the transparency of the chart image, do a Google search for place names, and set a home button which acts as a location bookmark (see Figure 8 ).



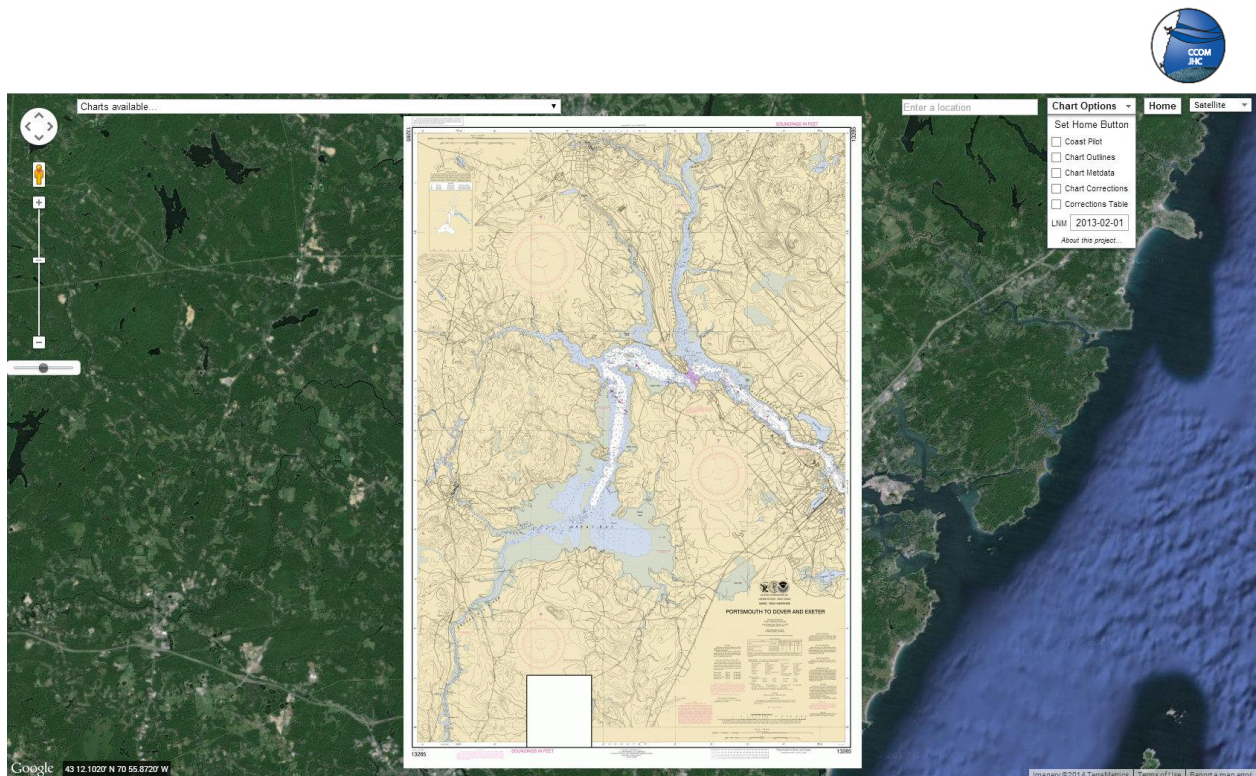


FIGURE 8. NCAPIUI (AKA. CHUM) ELEMENTS

## COMPONENTS OF THE NAUTICAL CHARTS APIUI

The NauticalChartsAPIUI is the parent class used to contain all the other user interface (UI) components and allow the interactivity between the components. To instantiate the NauticalChartsAPIUI class it just needs a Google Map as its argument. For example:

```
var ui = new NauticalChartsAPIUI(map);
```

There are four main classes in the Nautical Charts API that are directly linked to the NauticalChartAPIUI. Using the example above, they would be added to the ui variable with their associated options like this:

```
ui.homeButton = new HomeButton(buttonOptions);
```

The possible options are laid out in Figure 9.

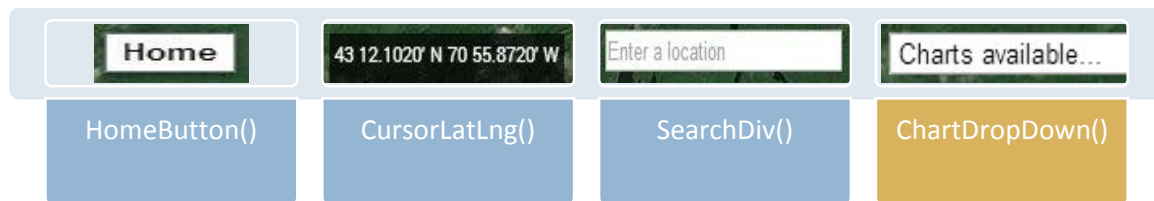


FIGURE 9. FOUR MAIN CLASSES DIRECTLY LINKED TO NCAPIUI

## COMPONENTS OF THE CHART DROP DOWN

The Chart Drop Down holds the “Chart Options” controls, any number of which can be added to the drop down, and can be ordered any way desired. To add the chart options to the drop down the following keywords are used: setHome, cp, outlines, metadata, markers, table, sinceDate and info. These keywords are linked to various Google style controls that can be reused, like buttons, checkboxes, links, and textboxes/date pickers (see Figure 11). These elements are all coded into the

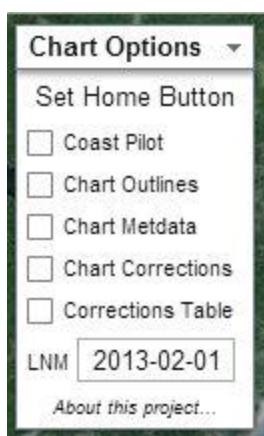


NCAPIUI and do not allow for any alterations except for the label name. Changing the functionality of these elements would entail changing the associated class code in the NCAPIUI.

The Opacity Control (Figure 10) is also associated with the chart even though the UI controls is not physically inside the Chart Drop Down Control (in other words, any element that needs a chart to function is placed within the ChartDropDown class).

For example:

```
Var cDD = ui.ChartDropDown(ui);  
cDD.opacityControl = new OpacityControl(100);
```



**FIGURE 11. CONTROLS  
THAT RESIDE INSIDE  
THE CHART DROP  
DOWN CONTROL**



**FIGURE 10. OPACITY CONTROL**

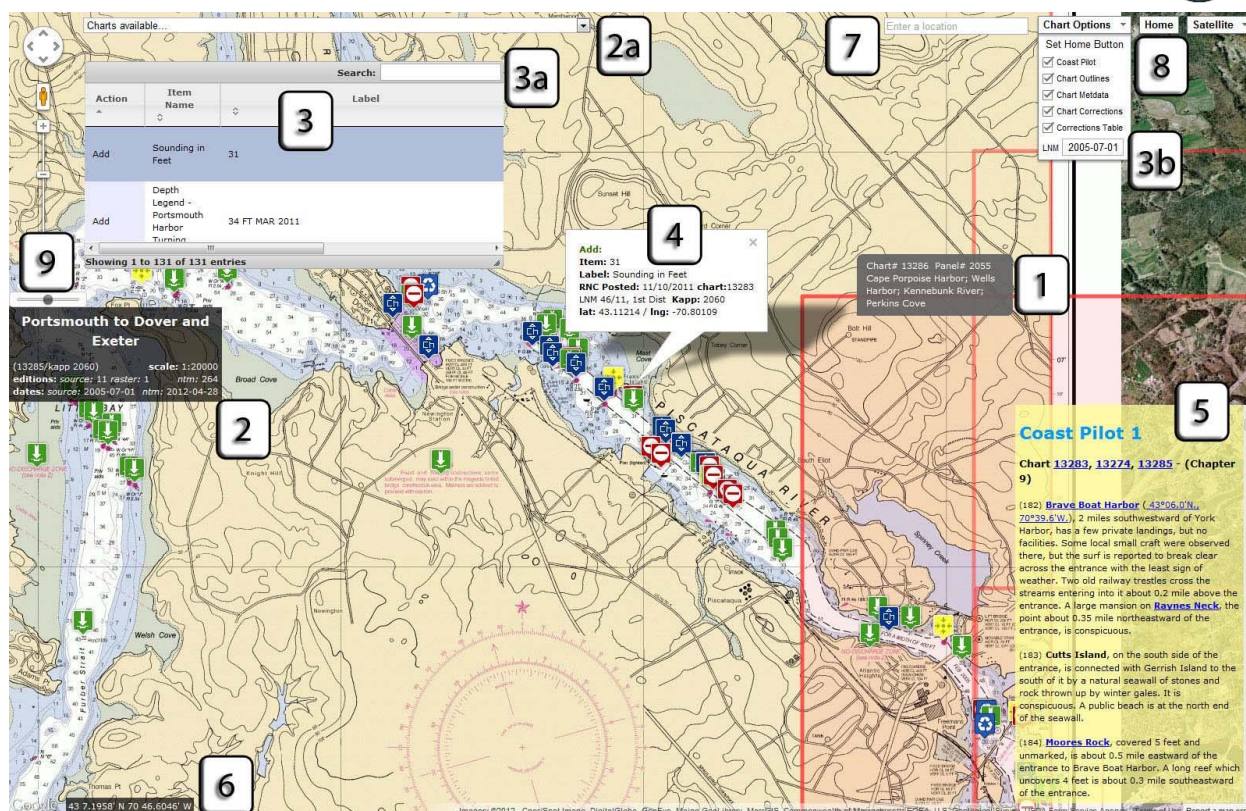
## COAST PILOT INTEGRATION

The NCAPIUI can also display the U.S. Coast Pilot for the chart selected in the chart drop-down box (see Figure 12, box 5 and 2a, respectively), with geo-referenced links to latitude/longitude locations for effortlessly finding points of interest on the chart. Instead of using the XML version of the Coast Pilot on the OCS website, the Coast Pilot data comes from the OCS U.S. Coast Pilot Search tool (found at: <http://www.nauticalcharts.noaa.gov/nsd/cpsearch.php>); this allows for retrieving HTML text related to a specific chart. The text is then processed using PHP, and items in “bold” style tags are sent to the Google Geocoding service. A successful return from the service yields a latitude/longitude position that is used to make a geo-referenced link in the Coast Pilot text displayed.

## CHUM™ OVERVIEW

The end result of what the NCAPIUI displays and allows the user to interact with is what is termed ChUM™.





**FIGURE 12. CHUM - THE GOOGLE MAPS MASH UP FOR THE LOCAL NOTICE TO MARINERS, CHART CATALOG AND COAST PILOT**

An overview of all the features ChUM™ offers, see Figure 12 for matching number reference:

1. Integrates NOAA chart catalogs
2. Integrates NOAA nautical charts
  - a. Ability to filter charts depending on viewport
3. Integrates critical chart corrections from OCS as a dynamic/interactive table
  - a. With the ability to filter data
  - b. With the ability to view historical data
4. Integrates critical chart corrections as geo-referenced markers within the context of a nautical chart
5. Integrates Coast Pilot w/Geo-coded places
6. Geo-referenced environment (lat/lng control)
7. Ability to search place names
8. Ability to save a location bookmark
9. Ability to set the chart transparency

## FUTURE IMPROVEMENTS

- Ideally, it would be nice to have some time to clean up the code...there is much to clean up! And upgrade various components to the newer versions (like the DataTables plug-in)
- This would be great on “seamless” tiled charts (instead of relying on UCSDs tile server), but a new mechanism would need to be in place for how the chart corrections are queried since it would no longer be associated with just one chart. It would be associated by boundary (user selected or the visible viewing area).



- It would be nice to have the critical chart corrections data in JSON (or XML) to remove the processing step of converting the text file to the JSON with PHP.
- It would be nice to have the Coast Pilot XML geo-tagged and features tagged...the Vislab did a mock-up of how this could be done using a features database and it made the cross-referencing to the chart very nice. (Instead of relying on Google Geo-coding service)
  - If continuing to use the Google Geo-coding service, need to verify the result instead of just taking the first item on the list.
- It would be nice to have ALL the notice to mariner information on the chart, everything that has a geo-reference, should be shown...items without a specific geo-reference should have a bounding area that would show up in a “news” items section when the bounded area is within the view port. (This would entail manually added geo-referencing when the notice is created.
- It would be nice to have this extended to include all the Notice to Mariners (from NGA) not just the Local Notice to Mariners.

## REFERENCES

SpryMedia Ltd ©. (2014, June 11). *DataTables*. Retrieved from DataTables: <http://datatables.net>

SpryMedia Ltd ©. (2014, 6 11). *DataTables CDN*. Retrieved from Data Tables Content Delivery Network: [cdn.datatables.net](http://cdn.datatables.net)

Coastal Observing Research and Development Center. (2014). *Nautical Charts API*. Retrieved from COASTAL OBSERVING RESEARCH AND DEVELOPMENT CENTER - Scripps Institute of Oceanography: <http://cordc.ucsd.edu/projects/charts/api/>

Google Geo Developers. (2009, May 27). *Announcing Google Maps API v3*. Retrieved from Google Geo Developers Blog: <http://googlegeodevelopers.blogspot.com/2009/05/announcinggoogle->

Google Maps Developer. (2014, March 11). *Google Maps JavaScript APIv3 - Map Types*. Retrieved from Google Developers: <https://developers.google.com/maps/documentation/javascript/maptypes?hl=fr#TileCoordinates>