BONUS BASMATI Deliverable 5.2: State of the Art Report on MultiChannel Map Applications

Dost, Ümit; Koski, Christian; Oksanen, Juha; Bonnevie, Ida Maria; Luhtala, Hanna

Publication date:
2018

Document Version
Publisher's PDF, also known as Version of record

BASMATI

Baltic Sea Maritime Spatial Planning for Sustainable Ecosystem Services

State of the Art Report on Multi-Channel Map Applications

Deliverable 5.2

BONUS

SCIENCE FOR A BETTER FUTURE OF THE BALTIC SEA REGION

BONUS BASMATI

State of the Art Report on Multi-Channel Map Applications

February 2018

Authors:

Ümit Dost, Christian Koski, Juha Oksanen, Finnish Geospatial Research Institute
Ida Reiter, Ålborg University
Hanna Luhtala, University of Turku

This report is a publicly accessible deliverable of the BONUS BASMATI project. The present work has been carried out within the project ‘Baltic Sea Maritime Spatial Planning for Sustainable Ecosystem Services (BONUS BASMATI)’, which has received funding from BONUS (art. 185), funded jointly by the EU, Innovation Fund Denmark, Swedish Research Council Formas, Academy of Finland, Latvian Ministry of Education and Science and Forschungszentrum Jülich GmbH (Germany).

Dost, Ü, Koski, C, Reiter, I, Luhtala, H, Oksanen, J, State of the Art Report on Multi-Channel Map Applications. BONUS BASMATI Deliverable 5.2, February 2017, www.bonusbasmati.eu

Contents

BONUS BASMATI project in short

BONUS call 2015:
Blue Baltic

Project coordinator:
Henning Sten Hansen, Aalborg University, Denmark

Project partners:
Aalborg University, Denmark (AAU)
Aarhus University, Denmark (AU)
Finnish Geospatial Research Institute, Finland (FGI)
Latvian Institute of Aquatic Ecology, Latvia (LIAE)
Leibniz Institute for Baltic Sea Research Warnemünde, Germany (IOW)
Nordregio, Sweden (Nordregio)
University of Turku, Finland (UTU)

Duration:
3 years, 7/2017 – 6/2020

Key theme addressed:
Theme 4.3 Maritime spatial planning from local to Baltic Sea region scale

Subthemes:
Theme 2.3 Integrated approaches to coastal management and Theme 4.1 Governance structures, policy performance and policy instruments

https://www.bonusportal.org/projects/blue_baltic_2017-2020

Abstract:
Maritime Spatial Planning (MSP) requires a spatially explicit framework for decision-making, and on that background the overall objective of BONUS BASMATI is to develop integrated and innovative solutions for MSP from the local to the Baltic Sea Region scale. This is to be realised through multi-level governance structures and interactive information technology aiming at developing an ecologically and socio-economically sound network of protected marine areas covering the Baltic Sea. Based on the results of former MSP projects, the BONUS BASMATI project sets out to analyse governance systems and their information needs regarding MSP in the Baltic Sea region in order to develop an operational, transnational model for MSP, while maintaining compliance with existing governance systems. It also develops methods and tools for assessments of different plan-proposals, while including spatially explicit pressures and effects on maritime ecosystem services in order to create the Baltic Explorer, which is a spatial decision support system (SDSS) for the Baltic Sea region to facilitate broad access to information. During the project running until 2020, new data will be produced and tested in assessments corresponding to policy goals. The data will support the combined analysis of the three elements of the concept of ecosystem services: the capacity, flow and benefit of provisioning, regulating and cultural services. A central aim of the project is to facilitate cross-border collaboration, and the project is carried out in close cooperation with relevant stakeholders in the BSR. The impact of the project will be facilitated and assessed in transnational case studies, where integrated solutions are required. The local scale will consist of case study areas in the South-West Baltic, the Latvian territorial and EEZ waters including open part of the Baltic Sea and the Gulf of Riga, and across the region, a pan-Baltic case study will be performed.

Summary

This document describes basic building stones for developing state of the art multi-channel map applications, such as the Baltic Explorer. By multi-channel we describe the system as being able to communicate information through multiple different devices, including personal computers, large touch screen devices, and smartphones. The Baltic Explorer that is to be developed in the BONUS BASMATI project, is a highly interactive, web-based spatial decision support system (SDSS), with functionality that includes geospatial data exploration, real-time cumulative impact assessment, co-location, and suitability analysis. By SDSS we mean a system designed for facilitating decision making processes based on complex spatial problems, and it provides an integrating framework for database management, geospatial analysis, visualisation, and the expert knowledge of decision makers.

Web mapping libraries provide frameworks for developing the client-side web map applications. The Baltic Explorer will feature a highly intuitive map-based interface, making web mapping libraries and application programming interfaces (APIs) ideal development tools for it. This report presents the results of two tests that were carried out for five different web mapping libraries (OpenLayers, Leaflet, ArcGIS API for JavaScript 3, ArcGIS API for JavaScript 4, Cesium). The tests provided insight on how the client-side of the Baltic Explorer could be constructed and highlighted some of the differences between the libraries. In the first test, the libraries were tested for integration with HELCOM ArcGIS REST API, a primary data source for the BONUS BASMATI project. In the second test, the APIs were tested for how their content adapts to smartphone screens. Modern web pages and applications use responsive web design (RWD) to make their content adapt to different sized screens. If RWD rules are included in a library, it can speed up the development of the Baltic Explorer and leaves more time for the development to focus on other issues.

The result of the two tests revealed that all five libraries were able to complete both tests, and as such, the tests were unable to exclude any of the libraries from consideration for the Baltic Explorer. However, some smaller differences were revealed in the tests. For example, unlike the other three libraries, OpenLayers and ArcGIS API for JavaScript 3 require extensions to become responsive to different screen sizes. RWD and web mapping libraries will be useful in developing the client-side of the Baltic Explorer. The chosen web mapping library will provide the backbone of the Baltic Explorers client-side and RWD will enable map interfaces to adapt to screens of different size, including computer monitors and large touch screens, both planned to be platforms for the Baltic Explorer. For smaller touch screen devices such as smartphones, additional design elements like larger buttons that are easy to use with fingers, might need to be implemented.

The next step in the development of the Baltic Explorer is to build the first prototype of the SDSS. In the upcoming months, more complex tests for the client-side web mapping libraries are needed in order to decide which library provides the best features for the Baltic Explorer. The Baltic Explorer is envisioned to be used in participatory settings with non-expert users, which emphasizes the need for good performance, and a user interface that is intuitive, interactive, and easy-to-use. Further testing of the mapping libraries will focus on these requirements. It is not yet decided if the Baltic Explorer needs to be capable of 3D visualisation, such as provided by ArcGIS API for JavaScript 4 and Cesium. If there is no need for a 3D map environment, these libraries might prove to be excessively big. Finally, while ArcGIS API for JavaScript 3 and 4 are free to use, they are not open source. Developers are thus not able see the source code, which to a degree limits the transparency of the system.

1 Introduction

According to a commonly applied definition, Maritime spatial planning (MSP) “is a public process of analysing and allocating the spatial and temporal distribution of human activities in marine areas to achieve ecological, economic, and social objectives that are usually specified through a political process” (Ehler et al., 2011). To reach its objectives, the planning process benefits from tools that can support decision making. Spatial decision support systems (SDSS) can provide the means to solve complex spatial analysis, engage stakeholders in planning processes, and communicate geographic information to decision makers. Despite there being many existing decision support tools for MSP, their use in real MSP processes is still limited (Pınarbaşı et al., 2017). SDSSs for MSP have sometimes been criticised for their poor accessibility caused by commercial licences as well as low usability due to high requirements on technical skills and GIS knowledge from their users. In addition, many tools focus on specific purposes to be used in certain stages of the planning process, which crucially limits their functionality if planners are forced to utilise a multitude of tools for covering all the stages of the process (Pınarbaşı et al., 2017). In the BONUS BASMATI project, a highly interactive SDSS, called the Baltic Explorer, will be developed. Its functionality will include geospatial data exploration, real-time cumulative impact assessment, co-location, and suitability analysis. The Baltic Explorer will address the weaknesses of current SDSS for MSP. It will be free and open source, making it both highly accessible and transparent. It will have an interactive, intuitive, and easy-to-use touch screen interface for non-expert users, to be used in collaborative MSP. The Baltic Explorer will be evaluated through two case studies that focus on site selections processes. The cases will consider the designation of marine protected areas in the Latvian coastal waters and investigation of opportunities for mussel farming in the south-western Baltic Sea. The third case study operates on a transboundary pan-Baltic scale. It will provide the development of the SDSS with additional insight to transboundary issues in MSP as well as understanding about the requirements of the stakeholders to participate in collaborative MSP.

This report describes basic building stones needed for developing an interactive multi-channel web map application such as the Baltic Explorer. By being a multi-channel application, the Baltic Explorer will be capable of communicating information through multiple different devices, for example desktop computers and large touch screen devices. Most web browsers interpret the same core web programming languages, and to a large extent the same code can be used for any of them. As the client-side of web applications are typically built using these languages web applications can be accessed through a large variety of different devices (smartphones, tablets, PCs) making them platform-independent (Farkas, 2017). Farkas (2017) uses the term ‘massive client’ to describe web clients that have advanced GIS characteristics, works in browsers, and thus needs to be browser-independent. However, the varying screen sizes of different devices that run web browsers present a challenge to the interface design of web applications. To help in adapting to different screen sizes, modern web pages and applications often use responsive web design (RWD) rules. RWD defines a set of rules that enables the content of web pages and applications to adapt to different screen sizes (Marcotte, 2015). The Baltic Explorer is envisioned to run on large touch screens and personal computers, and will possibly have additional functionality on smartphones in participatory settings. Therefore, RWD rules need to be applied, and both mouse interaction and touch screen gesture input methods need to be considered in its user interface design.

Each web-based GIS software needs to contain a server-side and a client-side component. The service-side services data while the client-side requests data (Farkas, 2017). The client-side of web map applications is often developed using pre-existing web mapping APIs (Application Programming Interfaces) and libraries. Light-weighted JavaScript-based APIs are part of a new web trend where web-based applications and data distribution standards enable users to create and view their own maps, an interactive web trend where the web is used for generating new data, referred to as ‘Web 2.0’ (Batty et al., 2010). In the development of the client-side of the Baltic Explorer, pre-existing client-side web mapping libraries will be used. The second chapter of this report discusses client-side web mapping libraries, RWD, and touch screen interface design. Five well-known and commonly used web mapping libraries (Leaflet, OpenLayers, ArcGIS API for JavaScript 3, ArcGIS API for JavaScript 4, Cesium) are introduced shortly. In the third chapter of this report, results from testing these five

libraries for consuming data from HELCOM's ArcGIS REST (Representational State Transfer) API, and for adopting to different sized screens through RWD are presented. The aim of these tests was to provide knowledge of the differences between these libraries and their suitability for being used as a client-side component for the Baltic Explorer.

2 Web mapping libraries, RWD, and touch screen interaction

2.1 Web mapping libraries

The client-side of web map applications is often built by combining the use of the core web technologies, Hypertext Markup Language (HTML), Cascading Style Sheets (CSS), and JavaScript, with client-side web mapping libraries. HTML is used for the basic structure of the page, CSS for styling, and JavaScript to add interactivity. Many of the current web mapping libraries are based on HTML and JavaScript (Veenendaal et al., 2017). The libraries support developers by providing ready-made classes and functions, discarding the need to write all the low-level logic that is required for interactive web map applications. This speeds up the development of the client-side of web map applications. APIs provide the interfaces for developers to use the libraries. Key elements of JavaScript web user interfaces are many options for user interactivity and fast loading time through loading-on-demand (Andreeva et al., 2012).

Below we give a short introduction to each of the five web mapping libraries that were tested for their ability to consume data from the HELCOM REST API and for how they behave when viewed on smartphone screens. The five libraries were chosen for the tests because they are some of the most well-known and commonly used free web mapping libraries that are available at the time. Important factors to consider in a performance analysis of web mapping libraries are general capabilities, platform dependency, documentation, support, and difficulty to use (Farkas, 2017).

2.1.1 OpenLayers

OpenLayers[1] is a client-side web mapping library based on JavaScript that allows placing of dynamic maps on web pages. Developers can add map tiles, vector data, and markers that can be loaded from any source by using its functions and classes. OpenLayers aims to improve the use of geographic information by providing extensive features. OpenLayers has good API documentation, good tutorials, and a moderate learning curve (Farkas, 2017). It has a wider community than Leaflet according to Khitrin (2017). Compared to Leaflet and Cesium (the two other open source libraries), OpenLayers supports a larger number of GIS features (Farkas, 2017). It is free and open source, released under the 2-clause BSD license (FreeBSD).

2.1.2 Leaflet

Leaflet[2] is a lightweight client-side web mapping library based on JavaScript, highly capable due to a high amount of competent third-party extensions. It has a modular structure, enabling it to reduce the library size to limit the functionality to basic needs for the task at hand, increasing page load speed (Khitrin, 2017). It aims to provide developers with an easy, simple, high performing, and mobile friendly web mapping library. It is easy to learn and has good API documentation and good tutorials**.** According to Farkas (2017), Leaflet should be a preferred choice when the project ambition is clear from the beginning (no scalability needed). It is free and open-source, released under a 2-clause BSD license license[3].

[1] OpenLayers home page: https://openlayers.org/
[2] Leaflet home page: http://leafletjs.com/
[3] The Leaflet BSD license: https://github.com/Leaflet/Leaflet/blob/master/LICENSE

2.1.3 ArcGIS API for JavaScript 3

ArcGIS API for JavaScript 3[4] is a free to use client-side web mapping API provided by ESRI. It integrates well with other ESRI solutions. It has extensive capabilities and documentation[5]. ArcGIS API for JavaScript 3 aims to maximize productivity and performance for developers. For this purpose, it uses a modular and component-based approach (Doman, 2015). Developers can determine, adapt, and embed necessary components according to their needs. ESRI can provide ArcGIS API for JavaScript developers with additional features through ESRI's developer's program and licenses.

2.1.4 ArcGIS API for JavaScript 4

ArcGIS API for JavaScript 4[6] is a free to use 3D and 2D web mapping API provided by ESRI. It aims to expand the capabilities of ArcGIS API for JavaScript 3, to support both 2D and 3D web mapping functionalities. However, it is not built on top of ArcGIS API for JavaScript 3, and instead has a different design (Pimpler, 2014). It also does not have all the capabilities of ArcGIS API for JavaScript 3. 3D applications developed with ArcGIS API for JavaScript 4 are powered by Web Scenes[7], which allows developers to visualise and analyse 2D and 3D data in a 3D environment.

2.1.5 Cesium

Cesium[8] is a virtual globe, capable of 3D visualisations due to its globe-like capabilities. It is an free and open source JavaScript library based on WebGL for hardware-accelerated graphics. Cesium developers describe their mission as: "Creating the leading 3D globe and map for static and time-dynamic content, with the best possible performance, precision, visual quality, platform support, community, and ease of use". However, it has a limited coverage of GIS features (Farkas, 2017) and supports only two coordinate systems (EPSG:4326, EPSG:3857). Cesium has good API documentation and good tutorials, but the number of developer questions asked on the internet is smaller than for web mapping libraries, and it has a steeper learning curve than Leaflet and OpenLayers (Farkas, 2017). Virtual globes are not necessarily larger than 2D web mapping libraries (Farkas, 2017), but Cesium (57 MB[9]) is significantly larger than Leaflet (38 KB[10]) and OpenLayers (3.1 MB[11]).

2.2 Responsive web design

Web browsers exist on many different devices, which can have vastly different screen sizes. RWD is a set of rules for developing web sites and applications in a manner that enables them to rescale their content according to the users screen resolution (Marcotte, 2015). Responsive web applications consist of four main technical elements: viewport, fluid grids, media queries, and flexible images. When the viewport HTML tag is used to set the visible area of a responsive web page or application, the page will automatically adapt to different sized screens[12].

A popular practice in web development is to build pages and applications using a set of grids or boxes. This means that all the content in the page is placed in a rectangle and the page content is divided into columns. Fluid grids allows to set the size of these boxes in flexible ways (Marcotte,

[4] Documentation for ArcGIS API for JavaScript 3: https://developers.arcgis.com/javascript/3/
[5] ArcGIS API for Javascript Overview: https://developers.arcgis.com/javascript/3/jshelp/
[6] Documentation for ArcGIS API for JavaScript 4: https://developers.arcgis.com/javascript/
[7] ArcGIS Online Help - Web Scenes: https://doc.arcgis.com/en/arcgis-online/reference/what-is-web-scene.htm
[8] Cesium home page: https://cesiumjs.org/
[9] Cesium download page: https://cesiumjs.org/downloads/
[10] Leaflet home page: http://leafletjs.com/index.html
[11] Size of unzipped library downloaded from OpenLayers download page: https://github.com/openlayers/openlayers/releases/tag/v4.6.4
[12] W3C tutorial for HTML Responsive Web Design: https://www.w3schools.com/html/html_responsive.asp

2015). By using fluid grids, every page element is sized in relative units like percentages rather than pixels (Gardner, 2011). A responsive grid-view divide the area usually into 12 columns and total width of area is 100%[13]. This area will also shrink and expand when the browser window is resized. Flexible images enable images to rescale according to screen resolution (Gardner, 2011).

Media queries enable web pages and applications to change any design elements based on the screen size (Marcotte, 2015). This type of behaviour is often needed, for example when content is also designed to be viewed on vertically held smartphones that have exceptionally narrow screens (Gardner, 2011).

2.3 Touch screen interaction and design guidelines

Touch screen devices have the capability to react to users touch gestures. They can be operated with a finger or touch pen (Tu et al., 2015) and they can provide more direct interaction between users and devices. Touch screens are usually operated on mobile devices, such as mobile phones and tablets, but also on large screens available at sale kiosks, cash stations, museums and more (Figure 1). In a typical use case, users of large touch screen in a public space are often not trained (non-expert users) and they would usually like to finish their tasks in a fast and simple way. However, the level of experience on touch screen devices vary among the users, and those who are more experienced tend to perform their tasks quicker than those with little or no experience (Samet et al., 2011).

Figure 1. *Highly interactive Tassu map application developed at the Finnish Geospatial Research Institute (FGI) supporting touch gestures and designed for public spaces, such as museums. (Picture: Julia Hautojärvi)*

Touch screen applications and traditional web applications have different capabilities and design guidelines. However, making a web application work on both of these devices can effectively be achieved by applying RWD rules. For example, it is important to remember that touch screen devices have limitations in hovering, highlighting and dragging events. In these circumstances, it is better to use click-to events when designing the application. On a touch screen device, it is not possible to get feedback about a page element when hovering over or highlighting. If it is necessary to use navigation bars, using it on the right side is also useful since many people are right handed. The size of the page elements and the spaces between them should consider the size of the finger and the device size.

[13] W3C tutorial for Responsive Web Design - Grid View: https://www.w3schools.com/cSS/css_rwd_grid.asp

Size of the page control elements should be large to increase the accuracy, ease of performance, and user satisfaction (Gao & Sun, 2005). Inputs should be given usually by click events instead of long text and number inputs as it is difficult to type on touch screens and it may be complicated to use more than one finger. Especially older people prefer click-to design, whereas younger users may favour manipulation gestures using multiple fingers (Gao & Sun, 2005). Application should be full screen and they should use bright colours to reduce problems induced by glare and fingerprints. It is also important to observe the Gestalt laws for grouping and arranging page elements (Chang & Nesbitt, 2006). The interface should be designed properly for selection options. When using single selection, radio buttons can be used. For multiple selections, check boxes are more appropriate. For multiple clicking, scroll, and spin buttons can be used.

3 Web mapping library tests

In this part of the report, results of testing five well known web mapping libraries for two important capabilities relevant for distributed web mapping will be presented. The first capability to be tested is the ability of the different libraries to consume ArcGIS REST API data. In the BONUS BASMATI project, mainly HELCOM data and maps services are used as data source, and HELCOM secretariat reveals all HELCOM datasets by providing open access to its data using ArcGIS REST API[14] and OGC standards[15]. A REST API is a standardised tiling service and an interface for client-server communication to enable the server-side to only send a simplified data subset to the client in order to consider the large sizes of data piles and keep data streams limited (Farkas, 2017). The ArcGIS REST API gives simple and open access to data sources and services that resides on ArcGIS Server. Each data resource exposed with the REST standard is given in the hierarchy of Unified Resource Locator (URL) end points. These resources are designed for developers to consume geospatial data in a logical and hierarchical structure[16].

The second capability to be tested is the responsiveness of each library to different screen sizes of different devices. In principle, RWD rules must be applied, not only to map content, but to the whole page content, to make it work nicely on different devices. Here, only the responsiveness of the map content to different devices will be tested by using Chrome built-in emulator and using the scripts for integrating each library with ArcGIS REST API.

The tests were carried out using the following versions of the web mapping libraries:

- OpenLayers 4.6
- Leaflet 1.2
- ArcGIS API for JavaScript 3.22
- ArcGIS API for JavaScript 4.5
- Cesium 1.42

3.1 Test 1: Integration with ArcGIS REST API

To test the five libraries' integration with ArcGIS REST API, five basic HTML scripts have been written; one script per library integration. Each script is interpreted into a map, when opened in a browser. The maps provide examples of the integration with each library and ArcGIS REST API, and they will be described and shown here. The HELCOM map layer 'Baltic Sea pressure and impact index'[17] is requested in all the examples.

[14] The directory for HELCOM's ArcGIS REST API to its map and data service is http://maps.helcom.fi/arcgis104/rest/services/MADS.
[15] HELCOM data and maps services: http://www.helcom.fi/baltic-sea-trends/data-maps
[16] To read more about ArcGIS REST API, see https://developers.arcgis.com/rest/
[17] Link to the Baltic Sea Pressure and Impact Index: http://maps.helcom.fi/arcgis104/rest/services/MADS/Pressures/MapServer

OpenLayers has direct support for using a dynamic ArcGIS REST map service. In this way, it is relatively simple to consume map and image services without plugins. In the test in this report, the index is easily viewed in an internet browser (e.g. Safari, Chrome etc.) by using a request built on OpenLayers' dynamic image class[18] and OpenLayers' imageArcGISRest[19] method. The requested index layer is shown to the user on top of the base map (Figure 2).

Figure 2. *The OpenLayers example for integrating the HELCOM Baltic Sea Pressure and impact Index layer.*

Leaflet is unlike OpenLayers not directly integrated with ArcGIS REST map services. It can, however, easily be integrated with the latter by using ESRI Leaflet plugin[20], which provides a lightweight set of tools for such an integration. In the test in this report, the Leaflet script defines a basemap[21] and uses ESRI Leaflet plugins dynamicMapLayer[22] method to request the index layer (Figure 3).

[18] Documentation for OpenLayers' dynamic image class: https://openlayers.org/en/latest/apidoc/ol.layer.Image.html
[19] Documentation for OpenLayers' imageArcgisRest method: https://openlayers.org/en/latest/apidoc/ol.source.ImageArcGISRest.html
[20] ESRI Leaflet plugin documentation: http://esri.github.io/esri-leaflet/
[21] ESRI Leaflet plugin's ESRI basemapLayer method documentation: http://esri.github.io/esri-leaflet/api-reference/layers/basemap-layer.html
[22] ESRI Leaflet plugin's ESRI dynamicMapLayer method documentation: http://esri.github.io/esri-leaflet/api-reference/layers/dynamic-map-layer.html

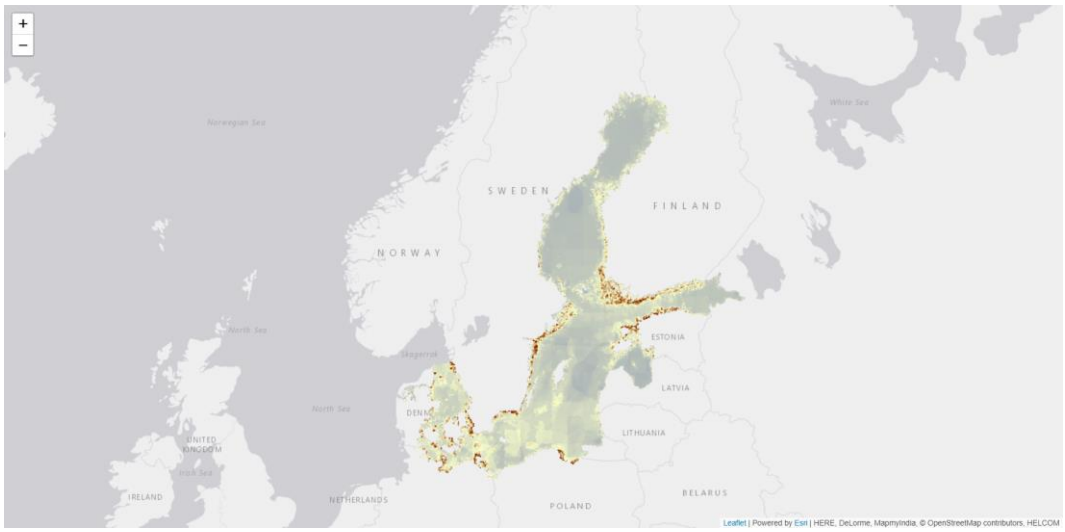

Figure 3. *The Leaflet example for integrating HELCOM Baltic Sea Pressure and impact Index layer.*

ArcGIS API for JavaScript has similarly to OpenLayers a direct integration with ArcGIS REST API. Through ArcGIS API for JavaScript 3's ArcGISDynamicMapServiceLayer[23] class, the index layer is requested in the test in this report (Figure 4).

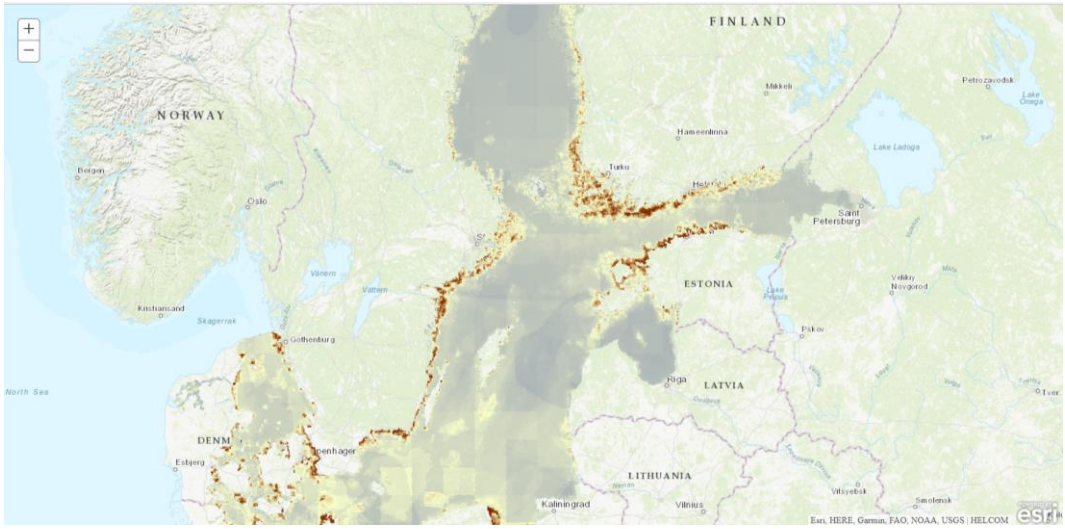

Figure 4. *The ArcGIS API for JavaScript 3 example for integrating HELCOM Baltic Sea Pressure and impact Index layer.*

Until now, the tests have all provided examples in 2D. With **ArcGIS API for JavaScript 4**, it is possible to request maps in 3D by choosing ArcGIS SceneView. Thus, a fourth test is created that enables the index layer to be viewed in 3D by using ArcGIS JavaScript API 4's mapImageLayer[24] class and its SceneView[25] class (Figure 5). The SceneView class enables options for the user to zoom, pan, and rotate the view. In this test, jQuery integration for switching on/off layers has also been added to enable users to click one of the four buttons below the map, which shifts between the different map layers. jQuery integration is possible for all the mentioned web mapping libraries. It is

[23] Documentation for ArcGIS API for JavaScript 3's ArcGISDynamicMapServiceLayer class: https://developers.arcgis.com/javascript/3/jsapi/arcgisdynamicmapservicelayer-amd.html
[24] Documentation for ArcGIS API for JavaScript 4's mapImageLayer class: https://developers.arcgis.com/javascript/latest/api-reference/esri-layers-MapImageLayer.html
[25] Documentation for ArcGIS API for JavaScript 4's SceneView class: https://developers.arcgis.com/javascript/latest/api-reference/esri-views-SceneView.html

a JavaScript library that can sometimes be used to solve browser incompatibilities issues (Andreeva et al., 2012), and it is known for being easy to learn (Farkas, 2017).

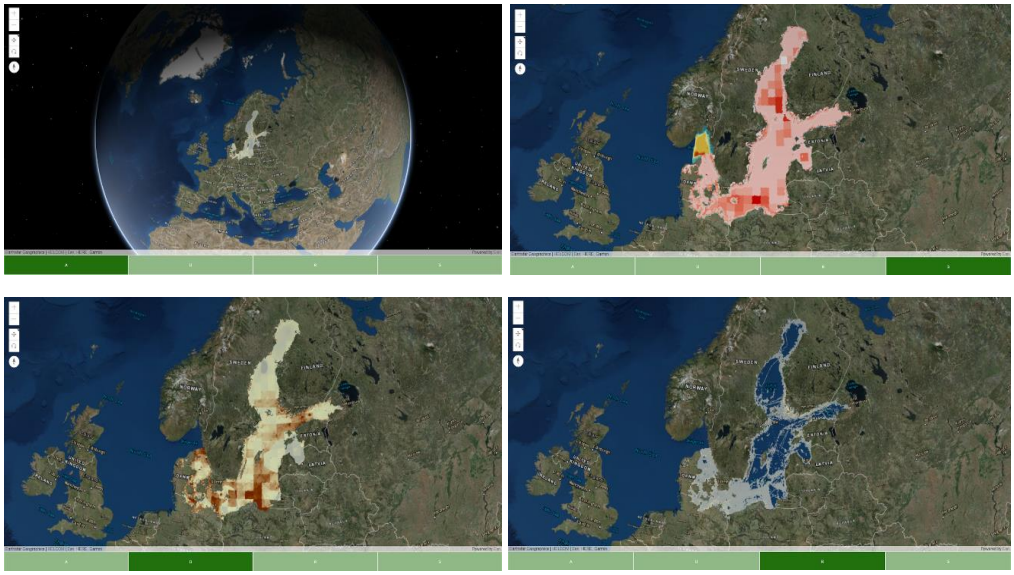

Figure 5. *ArcGIS API for JavaScript 4 example with buttons to switch between pressure layers.*

When using **Cesium** for 3D web mapping, it is possible to get a similar result as in the previous ArcGIS example (Figure 6). Cesium includes a set of 3D base maps where users can choose which one to use. Users can change the view between 3D, 2D, and Columbus view. Cesium includes built in search functionality. Cesium also has clear instructions to use the panning, zooming and rotating the view. Cesium was easily adapted to ArcGIS REST API by using Cesium's imageryLayers[26] class and its addImageryProvider method.

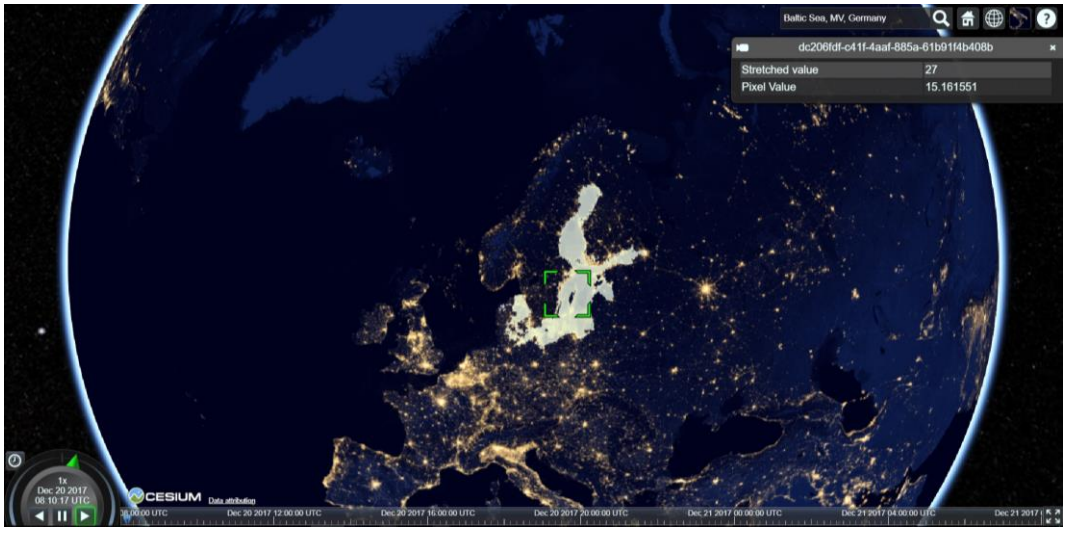

Figure 6. *The Cesium example for integrating HELCOM Baltic Sea Pressure and impact Index layer.*

The above tests have provided a way for each web mapping library to integrate ArcGIS REST API, which is relevant for Baltic Explorer that will need to request HELCOM data.

3.2 Test 2: Responsiveness

Responsiveness is important for a web map application to support different devices. Earlier in this

[26] Documentation for Cesium's imageryLayers class and addImageryProvider method: https://cesiumjs.org/tutorials/Imagery-Layers-Tutorial/

report, we described four different technical elements to implement responsiveness in web design. These elements can be applied to control different types of application content including images, footer, header, panel etc. Here, we are only testing responsiveness for web applications with a single content; a map. Thus, the only technical element of the four possible ones implemented for this purpose in the scripts, is a viewport: <*meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1, minimum-scale=1, user-scalable=no"*>. Fluid grids are not used, since we only work with one container/box and not different boxes/grids. Neither are media queries nor flexible images applied here in these demos.

For testing whether the applications demonstrated in section 4.1 are responsive or not, whether it works on different devices, it is possible to use a simulator or an emulator, when different devices are not available at hand. Emulators are able to simulate the screen size and certain characteristics of mobile devices[27]. To test map content responsiveness in this report, Chrome built in emulator[28] is used to emulate the performance of the above scripts on a mobile device while in reality using a personal computer.

When using **OpenLayers**, the width and height of CSS properties for the map content had to be set to 100%. In this way, it was possible to fit the content to the entire page. As seen in Figure 7 below, the map originally doesn't scale its content to fit into the entire page (left picture). When CSS properties are changed to full width and height, the content fits nicely to the mobile view and the whole content rescales (right picture). However, the navigation buttons stay small in both cases, which makes them difficult to use with fingers. To solve such an issue, custom buttons could be added.

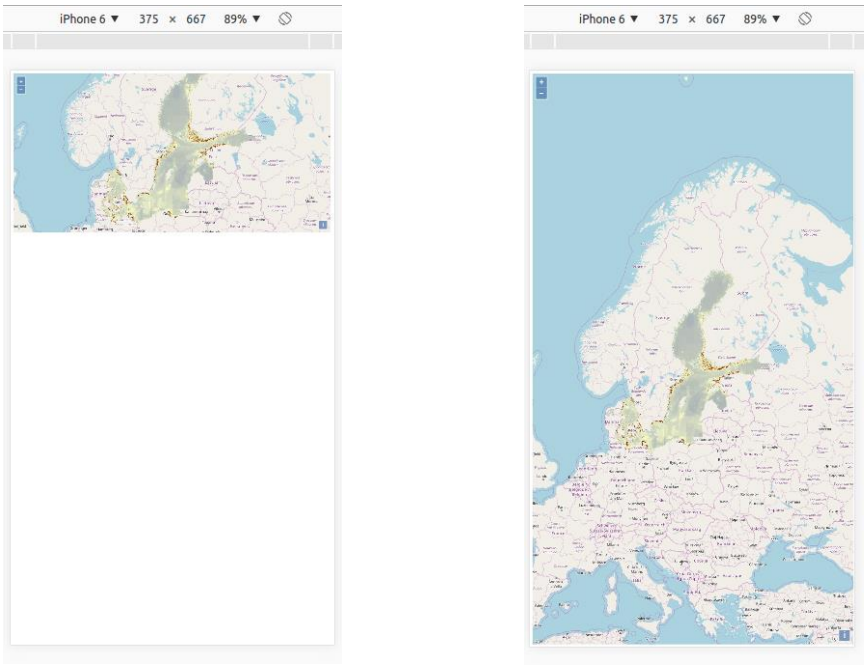

Figure 7. *OpenLayers' mobile view tested where the map doesn't fit the entire view (left) and where it fits properly after content rescaled to 100% (right).*

Leaflet is mobile friendly by default, which was revealed in its mobile device emulation. When scaling the screen size, the content still fits nicely on the screen. The map doesn't scale to view the whole map, but only views the content where an overlay map is present. Figure 8 below shows the Leaflet example in mobile view.

[27] A tutorial on testing responsiveness and device specific viewports with Chrome built-in emulator: https://developers.google.com/web/tools/chrome-devtools/device-mode/emulate-mobile-viewports

[28] A tutorial to use Chrome built-in emulator called Chrome Dev-Tools' Device Mode: https://developers.google.com/web/tools/chrome-devtools/device-mode/

Figure 8. *Leaflet mobile.*

ArcGIS API for JavaScript 3 must use another extension for making responsive maps. ESRI bootstrap map.js is a lightweight extension to blend ArcGIS API for JavaScript with Bootstrap in order to create responsive web map applications. When using this extension, maps resize and re-centre automatically as the size of the screen changes. Figure 9 shows how the ArcGIS API for JavaScript 3 map only starts to centre its map content and resize properly after ESRI Bootstrap Map[29] extension is used.

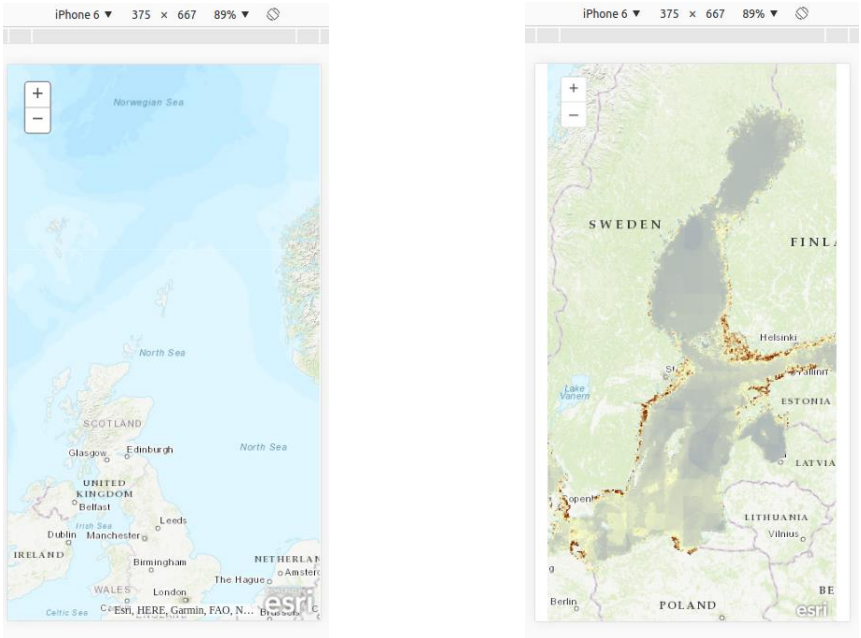

Figure 9. *The mobile view using ArcGIS API for JavaScript 3 without the bootstrap extension (left) and with the extension (right).*

Moving to the 3D web mapping libraries, they showed in the tests here to generally work nicely in mobile view. They fit the screen size, rescale map content, support touch gestures, and are usable without making any configurations. Figure 10 shows the **ArcGIS API for JavaScript 4** and **Cesium**

[29] Documentation for ESRI Bootstrap Map extension: https://github.com/Esri/bootstrap-map-js

tests in mobile view without any configurations.

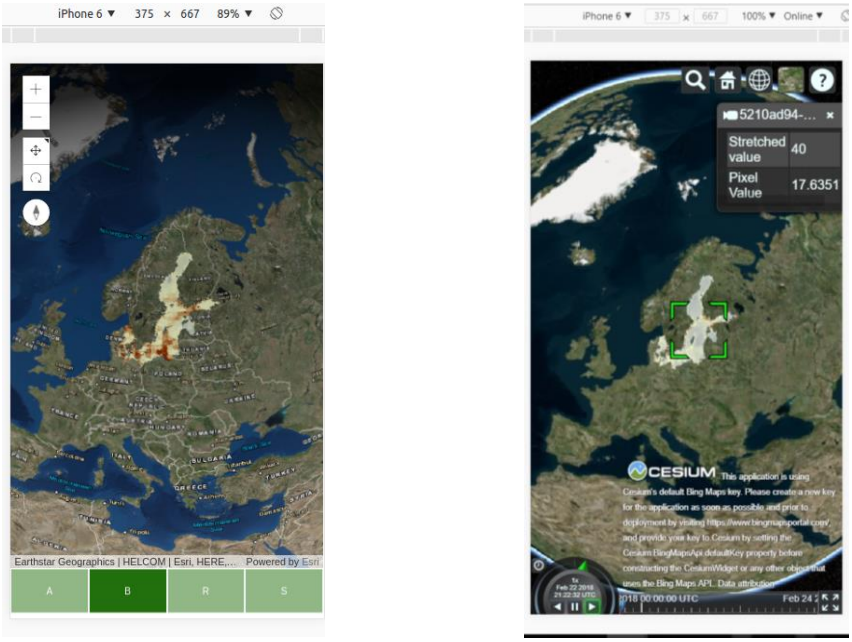

Figure 10. *The mobile view using ArcGIS API for JavaScript 4 (left) and Cesium (right).*

Summing up, no clear general best choice of web mapping library exists, which is similar to what Farkas (2017) concludes in his web mapping library review. However, some differences exist between the libraries (Farkas, 2017), also underlined by the tests in this report. The degree of map content responsiveness in the 2D web mapping libraries were generally limited in the tests in this report, except for Leaflet. Extra configurations were needed for OpenLayers and ArcGIS API for JavaScript 3 to ensure the map content to centre nicely. However, the 2D web mapping libraries do support touch gestures, which is necessary for a mobile view.

3D web mapping libraries such as ArcGIS API for JavaScript 4 and Cesium showed to be more responsive than OpenLayers and JavaScript API 3 in that the former scale their content automatically to match different sized screens. Instead of showing the centred map content such as all the 2D web mapping libraries, they scale to view the whole map, and let users zoom in and out.

4 Conclusions

This report has described basic building stones needed for developing an interactive multi-channel map application such as the Baltic Explorer. By testing five well-known and commonly used client-side web mapping libraries for two different properties, we managed to reveal some of their behavioural differences. For example, unlike the other three libraries, OpenLayers and ArcGIS API for JavaScript 3 require extensions to become responsive to different screen sizes. Despite these minor differences, all five libraries passed both tests (connecting to the HELCOM ArcGIS REST API and responding to different screen sizes). Based on these results, none of the five libraries can be excluded from consideration for the Baltic Explorer.

RWD and web mapping libraries will be useful in developing the client-side of the Baltic Explorer. The chosen web mapping library will provide the backbone of the Baltic Explorers client-side and RWD will enable map interfaces to adapt to screens of different sized devices, including computer monitors and large touch screens, both planned to be platforms for the Baltic Explorer. For smaller touch screen devices such as smartphones, additional design elements like larger buttons for fingers, might need to be implemented.

The next step in the development of the Baltic Explorer is to build the first prototype of the SDSS. In the upcoming months, more complex tests for the client-side web mapping libraries are needed in order to decide which library provides the best features for the Baltic Explorer. The Baltic Explorer is

envisioned to be used in participatory settings with non-expert users, which emphasizes the need for good performance, and a user interface that is intuitive, interactive, and easy-to-use. Further testing of the mapping libraries will focus on these requirements. It is not yet decided if the Baltic Explorer needs to be capable of 3D visualisation, such as provided by ArcGIS API for JavaScript 4 and Cesium. If there is no need for a 3D map environment, these libraries might prove to be excessively big. Finally, while ArcGIS API for JavaScript 3 and 4 are free to use, they are not open source. Developers are thus not able see the source code, which to a degree limits the transparency of the system.

References

Andreeva, J., Dzhunov, I., Karavakis, E., Kokoszkiewicz, L., Nowotka, M., Saiz, P., and Tuckett, D., 2012. Designing and developing portable large-scale JavaScript web applications within the Experiment Dashboard framework. *Journal of Physics*: Conference Series 396. International Conference on Computing in High Energy and Nuclear Physics 2012 (CHEP2012), pp. 1–11.

Batty, M., Hudson-Smith, A., Milton, R., and Crooks, A., 2010. Map mashups, Web 2.0 and the GIS revolution. *Annals of GIS,* 16(1), pp. 1–13.

Chang, D., and Nesbitt, K.V., 2006. Identifying Commonly-Used Gestalt Principles as a Design Framework for Multi-Sensory Displays. 2006 IEEE International Conference on Systems, Man, and Cybernetics, pp. 2452–2457.

Doman, K., 2015. *Mastering ArcGIS Server Development with JavaScript*. Packt Publishing Ltd.

Ehler, C., and Douvere, F., 2009. Marine spatial planning, a step-by-step approach towards ecosystem-based management. Intergovernmental Oceanographic Commission and Man and the Biosphere Programme. IOC Manual and Guides no. 53, ICAM Dossier no. 6. Paris: UNESCO.

Farkas, G., 2017. Applicability of open-source web mapping libraries for building massive Web GIS clients. *Journal of Geographical Systems*, *19*(3), pp. 273–295.

Gao, Q., and Sun, Q., 2015. Examining the Usability of Touch Screen Gestures for Older and Younger Adults. Human Factors, 57, pp. 835–863.

Gardner, B.S., 2011. Responsive web design: Enriching the user experience. *Sigma Journal: Inside the Digital Ecosystem*, *11*(1), pp.13–19.

Khitrin, M.O., 2017. Comparison of JavaScript Libraries for Web-Cartography. Bulletin of the South Ural State University. Ser. Computer Technologies, Automatic Control and Radioelectronics, 17(3), pp. 147–152.

Marcotte, E., 2015. *Responsive web design: A book apart n° 4*. Editions Eyrolles.

Pimpler, E., 2014. *Building web and mobile ArcGIS server applications with JavaScript: master the ArcGIS API for JavaScript, and build exciting, custom web and mobile GIS applications with the ArcGIS server*. Packt Publishing Ltd., UK.

Pınarbaşı, K., Galparsoro, I., Borja, Á., Stelzenmüller, V., Ehler, C. N., and Gimpel, A., 2017. Decision support tools in marine spatial planning: Present applications, gaps and future perspectives. *Marine Policy*, *83*, pp. 83–91.

Samet, H., Teitler, B.E., Adelfio, M.D., and Lieberman, M.D., 2011. Adapting a Map Query Interface for a Gesturing Touch Screen Interface. Proceedings of the 20th international conference companion on World Wide Web, pp. 257–260.

Tu, H., Ren, X., and Zhai, S., 2015. Differences and Similarities between Finger and Pen Stroke Gestures on Stationary and Mobile devices. ACM Transactions on Computer-Human Interaction 22, pp. 22–39.

Veenendaal, B., Brovelli, M.A., and Li, S., 2017. Review of Web Mapping: Eras, Trends and Directions. *ISPRS International Journal of Geo-Information*, *6*(10), 317.


AALBORG UNIVERSITY
DENMARK


AARHUS
UNIVERSITY


NLS
FINNISH GEOSPATIAL
RESEARCH INSTITUTE
FGI


LATVIJAS
HIDROEKOLOĢIJAS
INSTITŪTS


Leibniz Institute for
Baltic Sea Research
Warnemünde
iow


Nordregio


Turun yliopisto
University of Turku