UNESCO-IHE
Institute for Water Education

Upande

SNIEAU

MERPMEDER

DIRECTION GENERALE DE L'EAU

INSTITUT NATIONAL DE L'EAU - UNIVERSITE D'ABOMEY-CALAVI

UAC

AQUA OMNES USUS UNIVERSE

Port'Ability

PORTEUR DE SAVOIR-FAIRE

A GUIDE TO OPENLAYERS V3

Objectives

In this exercise you will learn the how to:

- Create a web map with HTML5 and JavaScript (OpenLayers v3)
- Add spatial content (maps and web map services to a website with OpenLayers 3
- Add controls for manipulating the spatial content on a website

Introduction

Web map applications are websites with mapping content in them. Web map application is a website — this means it is written in a language interpretable to the web browser this language is called the HyperText Markup Language (HTML). JavaScript language is used to allow user interaction in websites.

Web map applications are usually targeted to a specific user group which determines the selection of the spatial content itself — i.e. what kind of spatial data and in which data format is used. We will use the OpenLayers JavaScript library to put spatial content to our web map application

Basic Concepts

HTML 5

What is HTML?
HTML is a markup language for describing web documents (web pages).

- HTML stands for Hyper Text Markup Language
- A markup language is a set of markup tags
- HTML documents are described by HTML tags
- Each HTML tag describes different document content

```
<!doctype html>
<html>
```

```
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="URL" type="text/css">
<style> </style>
<title></title>
<iframe src="URL"></iframe>
</body>
</html>
```

Example Explained

- The <!DOCTYPE html> declaration defines this document to be HTML5
- The text between <html> and </html> describes an HTML document
- The text between <head> and </head> provides information about the document
- The link allows reference to other web pages and library sources: styles and scripts
- The text between <title> and </title> provides a title for the document
- The text between <body> and </body> describes the visible page content
- An iframe is used to display a web page within a web page.

Map

The core component of OpenLayers 3 is the map (ol.Map). It is rendered to a target container (e.g. a div element on the web page that contains the map). All map properties can either be configured at construction time, or by using setter methods, e.g. setTarget().

```
<div id="map" style="width: 100%, height: 400px"></div>
<script>
  var map = new ol.Map({target: 'map'});
</script>
```

View

ol.Map is not responsible for things like center, zoom level and projection of the map. Instead, these are properties of an ol.View instance.

```
 map.setView(new ol.View({
    center: [0, 0],
    zoom: 2
  }));
```

An ol.View also has a projection. The projection determines the coordinate system of the center and the units for map resolution calculations. If not specified (like in the above snippet), the default projection is Spherical Mercator (EPSG:3857), with meters as map units.

Source

To get remote data for a layer, OpenLayers 3 uses ol.source.Source subclasses. These are available for free and commercial map tile services like OpenStreetMap or Bing, for OGC sources like WMS or WMTS, and for vector data in formats like GeoJSON or KML.

```
 var osmSource = new ol.source.OSM();
```

Layer

A layer is a visual representation of data from a source. OpenLayers 3 has three basic types of layers:

- ol.layer.Tile
- ol.layer.Image
- ol.layer.Vector.

ol.layer.Tile is for layer sources that provide pre-rendered, tiled images in grids that are organized by zoom levels for specific resolutions.
ol.layer.Image is for server rendered images that are available for arbitrary extents and resolutions.
ol.layer.Vector is for vector data that is rendered client-side.

```
 var osmLayer = new ol.layer.Tile({source: osmSource});
  map.addLayer(osmLayer);
```

Putting it all together

The above snippets can be conflated to a self contained map configuration with view and layers:

```
<div id="map" style="width: 100%, height: 400px"></div>
<script>
  new ol.Map({
    layers: [
      new ol.layer.Tile({source: new ol.source.OSM()})
    ],
    view: new ol.View({
      center: [0, 0],
      zoom: 2
    }),
    target: 'map'
  });
</script>
```

Prerequisites

- Download the exercise files from **Download the SNIEAU Training: OpenLayers v3 Tutorial document**
- Create a main folder OLTutorial
- Create subfolders in OLTutorial: **MapIframe, Task1, Task2, Task3, Task4 & SimpeWebMap**

Publishing Map Iframe

Listing:**MapIframe/iframe.html**

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="http://openlayers.org/en/v3.8.2/css/ol.css" type="text/css">
<style>
#map {
height: 400px;
width: 400px;
}
</style>
<title>SNIEAU Iframe</title>
<iframe src="URL"></iframe>
</body>
</html>
```

URL: Replace with a published map from GeoNode instance
Example:

```
<iframe style="border: none;" height="400" width="600"
src="http://192.81.212.100/maps/22/embed"></iframe>
```

Exercise:MapIframe

- Publish a GeoNode map and replace **URL.**
- Change **width** and **height** of the iframe

Task 1

Create a new file and save it as **map.html** somewhere on your drive. Copy in the contents in listing 1 to the new file, save and open it .html in a web browser.

Listing 1:**Task1/map.html**

```
<!doctype html>
<html>
<head>
<meta charset="UTF-8">
<link rel="stylesheet" href="http://openlayers.org/en/v3.8.2/css/ol.css" type="text/css">
<style>
#map {
height: 400px;
width: 400px;
}
</style>
<title>Example OpenLayers3 page</title>
<script src="http://openlayers.org/en/v3.8.2/build/ol.js" ype="text/javascript"></script>
</head>
<body>
<h1>My Map</h1>
```

```
<div id="map"></div>
<script type="text/javascript">
var map = new ol.Map({
target: 'map',
layers: [
new ol.layer.Tile({
source: new ol.source.OSM({layer: 'osm'})
})
],
view: new ol.View({
center: [254031,6254016],
zoom: 16
})
});
</script>
</body>
</html>
```

To include a map on a web page we will need three things:

1. Include the OpenLayers library, which allows us to put spatial content on a web-
page:
This line includes the OpenLayers library to the webpage:

```
<script src="http://openlayers.org/en/v3.8.2/build/ol.js" type="text/javascript"></script>
```

.
The first part is to include the JavaScript library. Technically, you could put the reference to a JavaScript library anywhere else in the .html document.

2. Creating a HTML container for our map:
First, we define a style sheet for our map — we will use the standard cascading
style sheet (CSS) defined in the OpenLayers and we include the link to this css:

```
<link rel="stylesheet" href="http://openlayers.org/en/v3.8.2/css/ol.css" type="text/css">
```

in the <head> of our document. <div> HTML holding a map is created with <div id="map" class="map"></div>.
Through this <div> the map properties like width, height and border can be controlled through CSS. Our map is **400 pixels high and 400 pixels wide**, and this definition is included
in the <head> of the .html document as well.

```
<style>
#map {
height: 400px;
width: 400px;
}
</style>
```

NOTE: The map to be full screen, the width would be 100%
.

3. Create a simple map with JavaScript by writing a script specifying the map details:
The next step in generating our map is to include some initialization code. In our case, we have included a <script> element at the bottom of our document <body> to do the work:

```
 <h1>My Map</h1>
<div id="map"></div>
<script type="text/javascript">
var map = new ol.Map({

target: 'map',
layers: [
new ol.layer.Tile({
source: new ol.source.OSM({layer: 'sat'})
})
],
view: new ol.View({
center: [254031,6254016],
zoom: 16
})
});
</script>
```

With this JavaScript code, a map object is created with a layer from the OSM
tile server zoomed on the Benin. Let's look closer at our script:

The following line:

```
var map = new ol.Map({ ... });
```

creates an OpenLayers Map object. Just by itself, this does nothing since there is no layers or interaction attached to it.

We attach the map object to the <div>, with <div id="map"> the map object takes a target into arguments. The value is the id of the <div>: target: 'map'

The layers: [...] array is used to define the list of layers available in the map. The first and only layer right now is a tiled layer:

```
layers: [new ol.layer.Tile({
source: new ol.source.OSM({layer: 'sat'})
})
]
```

Layers in OpenLayers 3 are defined with a type (Image, Tile or Vector) which contains a source. The source is the protocol used to get the map tiles. You can consult the list of available layer sources here: http://openlayers.org/en/v3.8.2/apidoc/ol.source.html

The next part of the Map object is the View. The view allows specifying the center, resolution, and rotation of the map. The simplest way to define a view is to define a center point and a zoom level. Note that zoom level 0 is zoomed out.

```
view: new ol.View({
center: [0,0],
zoom: 16
})
```

```
Question?
What is the coordinate for Contonou, Benin ?
E.g. 6.6010193782859 N, 2.241625076858 E
```

Note that the center specified is in coordinates defined in the spatial reference system called Pseudo-Mercator (EPSG: 3857), which is native to OpenLayers.

3 Adding external web map services to a map
Now we learn how to add more layers to it. In previous exercises you published your own web map services, and now let's see how to
add these services to our map. For this exercise we will create a new web page.

Exercise 1:Task1/map.html

- Get the coordinates of Benin and **centre** the map. Use map.html
- Increase width & height probably to full screen: Example **width: 100%; height: 100%**

Task 2:

Create a new file and save it as map2.html somewhere on your drive. Copy in the contents in listing 2 to the new file.

Listing 2:**Task2/map.html**

```
<!doctype html>
<!-- Simple example of Web Mapping using HTML and OpenLayers 3/JavaScript-->
<html>
<head>
<meta charset="UTF-8">
<!-- definition of the styles -->
<link rel="stylesheet" href="http://openlayers.org/en/v3.8.2/css/ol.css" type="text/css">
<style>
.map {
height: 500px;
width: 500px;
}
</style>
<!-- JavaScript libraries used in the application -->
<script src="http://openlayers.org/en/v3.8.2/build/ol.js" type="text/javascript"></script>
<!-- My own OpenLayers 3/JavaScript code -->
<script src="layers.js" type="text/javascript"></script>
```

```
<title>OpenLayers 3 Example</title>
</head>
<body onload="init()"> <!-- init() is a JavaScript function defined in a separate file -->
<h2>My Map of Benin</h2>
<div id="map" class="map" style="float:left;"></div>
</body>
</html>
```

Observe the inclusion of layers.js script in the head (find **<script src="layers.js" type="text/javascript"></script>** line in the <head of your document).

Exercise 2:

- Change the title to represent the name of resources or utilities to be displayed
- Change map style to float on the right
  ```
  <div id="map" class="map" style="float:right;"></div>
  ```
- Change the dimensions of the map: width & height

Task 3:

Create a new file and save it as **layers.js** in the same directory where your **Task2/map.html** 'lives'.
In the following tasks we will step-by-step build our layers.js. Adding a layer is defined through several steps:
• definition of layer's a data source,
• definition of the 'initial behavior' of the map, and
• addition of the layer to the map.

Task 3.1:
Defining the layer's data source:
In our example, we will define the OpenStreetMap as the data source for the layer.
Add OSM's 'osm' data as the source for your layer. The code for this is the following:

```
var source1 = new ol.source.OSM({layer: 'osm'});
var osmlayer = new ol.layer.Tile({source: source1});
```

Task 3.2:
Building the init() function:
Add the following code to your .js file to define the 'initial behavior' of the map on a webpage:

```
function init() {
var view = new ol.View({
center: ol.proj.transform([100.50, 13.96], 'EPSG:4326', 'EPSG:3857'),
zoom: 5
});
var map = new ol.Map({
```

```
target: 'map',
controls: ol.control.defaults().extend([
new ol.control.ScaleLine({
units: 'metric'
})
]),
view: view,
});
map.addLayer(osmlayer);
}
```

Finally we need to write a script for adding our layer to the map. You can do this by adding **map.addLayer(osmlayer);** script to the **init() function.** The complete annotated code for creation a simple map containing one layer (the OpenStreetMap) centered to Benin is here:

Listing 3.2: **Task3/layers.js**

```
// definition of the layer source
var source1 = new ol.source.OSM({layer: 'osm'});
var osmlayer = new ol.layer.TileWMS({source: source1});
// init() function referenced in the <body onload=init()... of the application's HTML code

function init() {
var view = new ol.View({
center: ol.proj.transform([100.5, 13.96], 'EPSG:4326', 'EPSG:3857'),
zoom: 5
});
var map = new ol.Map({ // map as a composition of the view and controls
target: 'map', // link to the HTML object in which the map should be displayed
controls: ol.control.defaults().extend([
new ol.control.ScaleLine({
units: 'metric'
})
]),
view: view,
});
// adding layers to the map
map.addLayer(osmlayer);
}
```

Let’s now add more layers — once we are in zoomed to Benin.

Task 3.4:
Create a new layer with Benin’s cities:
• the URL to the service is: http://192.81.212.100/geoserver/**<your workspace>**/wms,
NOTE: You can change the url to point to your local GeoNode instance OR
http://www.snieau.bj/

• the name of the layer is: **<your workspace>**:**<layer>** Example **geonode:benin_cities**
• the server type is: **geoserver**
Add the following code to your .js file (NOTE: replace '<your workspace>' with the name of your own workspace on GeoServer where you published your web services):

```
var source2 = new ol.source.TileWMS({
url: 'http://192.81.212.100/geoserver/<your workspace>/wms',
params: {'LAYERS': '<your workspace>:benin_cities'},
serverType: 'geoserver',
});
var benin_cities = new ol.layer.Tile({
extent:[-1.46821054810656,6.3603727406018,2.94001664084976,11.1303658233031],
source: source2
});
```

NOTE: The extent defined for the layer is the spatial extent in the spatial reference system you defined for your web map service on GeoServer. You can find these values in your **WMS Capabilities document**. And of course, do not forget to add your new layer to the map.

Task 3.5:
Include **map.addLayer(benin_cities);** script into the definition of your init() function (after you added the OSM layer to the map).

Spatial reference systems in OpenLayers 3

OpenLayers 3 default spatial reference systems (SRS) is the WGS84/Pseudo-Mercator (EPSG:3857) (also known as the Spherical Mercator). This is the projected coordinate system used for rendering maps in Google Maps, OpenStreetMap, etc.

OpenLayers 3 has two methods allowing transformation between reference systems: http://epsg.io/3857

1. Ol.proj.transform for transforming coordinates, and
2. Ol.proj.transformExtent for transforming spatial extent

Task 3.6:
Re-write your layer's definition as follows:

```
extent:ol.proj.transformExtent([-
1.46821054810656,6.3603727406018,2.94001664084976,11.1303658233031], 'EPSG:4326', 'EPSG:3857',
```

This addition completes our forest layer definition to the following code:

```
var source2 = new ol.source.TileWMS({
url: 'http://localhost:8080/geoserver/<your workspace>/wms',
params: {'LAYERS': '<your workspace>:benin_cities'},
serverType: 'geoserver',
});
var benin_cities = new ol.layer.Tile({
extent: ol.proj.transformExtent ([-
1.46821054810656,6.3603727406018,2.94001664084976,11.1303658233031],'EPSG:4326', 'EPSG:3857'),
source: source2
});
```

When you refresh your map now, you should now see a cities map layer on top of the OpenStreetMap layer.

Exercise 3.6:

- Add a wms layer (any) Note: Extent of the layer
- Add a raster layer:**geonode_mnt_benin:**
- Add two layers: **raster and vector**
- Interchange wms layers: raster and vector

Task 4:

Let's see how we can add a map legend.

Adding legend to a map

OpenLayers 3 provides some of the controls for interaction with the dynamic spatial content, such as the zoom buttons, that come by default with the 'map' object and the 'scaleline' which can be added manually.
Fortunately, our layers are Web Map Services which, are able to provide their own legend — the **GetLegendGraphic service** operation helps doing this, it returns an image with the layer's legend.

(NOTE: replace<your workspace>with the your own workspace name): **geonode**

```
<div id="legend" style ="width :250 px; height :200 px;"> <!--placeholder for legend-->
<img src=
"http://192.81.212.100:8080/geoserver/geonode/wms?service=WMS&version=1.1.0&request=GetLegendG
raphic&layer=<your workspace>:benin_cities&format=image/png&width=20&height=20"><br/>
</div > <!-- GetLegendGraphic request to the WMS-->
```

Exercise 4.0:

- Add a legend for geonode_mnt_benin
- Add another geonode layer & its legend

Task 4.1:
OL3 OL3 We now have have two layers in our map, however, we cannot decide which one to see and which one not to see — can we fix this too?

Setting the visibility of layers:
In your **Task4/layer_visibility.js** file (at the end of the file) write a new function defining the initial status of the layer's visibility — the code for this function is available here:

```
function layerVis (value) { // function invoking the initial status of the layers' visibility
if (value == "osmlayer" ) {
osmlayer.setVisible(document.getElementById("1").checked);
}
else if (value == "benin_cities") {
benin_citieslayer.setVisible(document.getElementById("2").checked);
}
}
```

In your **Task4/map.html** code create a new placeholder for layer visibility switcher — the example
code is here:

```
<div style="padding:10px;">
<h2> Layer visibility switcher: </h2> <!-- a checkbox activating a function 'layerVis' defined
in a separate .js file (in our case, in the 'layers.js' file) -->
<input id="1" value="osmlayer" type="checkbox" onchange="layerVis(this.value)" checked/>
OSM-OpenStreetMap<br>
<input id="2" value="benin_cities" type="checkbox" onchange="layerVis(this.value)" checked/>
Benin Cities<br>
</div>
```

Exercise 4.1:

- Add another vector layer Example: **benin_rivers** to **visibility switcher**

Task 4.2:
Our map is almost ready, we want to include one more nice tool which will enhance our map's usability. This is the tool will provide additional information about the spatial data displayed on the map and will use the **GetFeatureInfo** operation every web map service offers to write it.

OpenLayers 3 supports this operation as well, and here is how we can use it:

- we need to include a function in our .js file allowing queries to the feature information related to our layers, and

- we need to create a placeholder in our .html file for displaying this information.

Write a new function in your **Task4/layer_info.js** file for retrieving the feature information from a layer. This code will be part of the init() function — insert the code below your view definition:

```
map.on('singleclick', function(evt) { // a click on a map allowing displaying non-spatial
attributes of the feature document.getElementById('info').innerHTML ='';
var viewResolution = /** @type {number} */ (view.getResolution());
var url = source2.getGetFeatureInfoUrl(evt.coordinate, viewResolution, 'EPSG:3857',
{'INFO_FORMAT':'text/html'}
);
if (url) {
document.getElementById('info').innerHTML =
'<iframe with="300px" height="100px" src="' + url + '"></iframe>';
}
}
);
```

Create a placeholder in your .html file for displaying features information — the code for this is here:

```
<h5> Click at the map to see additional information... </h5>
<div id="info">
<!-- Attributes of features will be displayed here -->
</div>
```

Exercise 4.2: Competition

1. Check “**OLTutorial/SimpleWebMap”** folder
2. Open in a browser the **map.html** file
3. Open **map.html** using leafpad/notepad (Any text editor)
    a. Add own map to the tab menu by creating your own **.html** file and adding replace it with **“#”**
    ```
    <li>
            <a href="#">Map 2</a>
    </li>
    <li>
            <a href="#">Map 3</a>
    </li>
    ```
4. Using own datasets:
    a. Upload datasets
    b. Style the datasets
    c. Publish maps
    d. Use map iframe & wms to add layers to your **SimpleWebMap/map.html**

SUMMARY

In this exercise we practiced creating simple Web map application with HTML5 and OpenLayers 3 and you should now be able to:

- create a simple website with HTML and JavaScript;
- add spatial content (maps and web map services) to a website with OpenLayers 3;
- add controls for manipulating the spatial content on a website;

REFERENCE

1. http://openlayers.org/en/v3.1.1/doc/quickstart.html
2. http://openlayers.org/ol3-workshop/basics/map.html