



JavaScript

Martin DELSINNE

Université Paris 8 - Master G2M

<https://github.com/mdelsinne/masterG2M-P8>

—
gfi.world

JavaScript

- I. Introduction
- II. Préparer son environnement de développement
Visual Studio Code
- III. Débuter sur OpenLayers
Nouvelle carte ol.Map
Ajouter des éléments à la carte ol.control
Ajouter des couches à la carte ol.layer
Sauvegarder ses sources sur GitHub
- IV. Découverte de l'API ArcGIS for JavaScript v4
2D / 3D et ajout de widgets

I. Introduction

› JavaScript : Langage de programmation web

› Inventé en 1995 ([Brendan Eich](#), co-fondateur de *Mozilla*)

- › Rendre dynamique les pages Web HTML

› Standardisé en 1997 : **ECMAScript**,

- › Variable non typée

- › Sensible à la casse

- › Enrichir les pages + Interactivité, contribue grandement au « web 2.0 »

› Aujourd'hui : **ES6** (JavaScript 2015)

- › Ajout de lib standard, nouvelle syntaxe et code plus lisible

- › À Lire : <http://exploringjs.com/es6/index.html>

```
1 var myVariable;
```



› Avec l'explosion du web, le javascript se généralise !

› Client : le code JS s'exécute sur le navigateur

- › Navigateur web (*Chrome, IE, Firefox, etc...*) - **WebResponsive**

› Smartphones et objets connectés :

- › iOS, Android ou Windows Phone : applications mobiles **multi-plateformes**



› **Node.js** : *nous reviendrons dessus plus en détail ultérieurement...*

- › applications rapides : moteur V8 de *Google*


- › JavaScript coté « serveur » !

I. Introduction

> APIs :

- >  **OpenLayers** - <https://openlayers.org/>
 - > Open-Source, communauté importante
 - > v3 : répandu, plus accessible
 - > v5 : dernière version v5, plus complexe mais plus rapide
 - > nécessite nodeJS
- > **ArcGIS API for JavaScript** - <https://developers.arcgis.com/javascript/>
 - > Version API 4.10 : 3D web apps – Arcgis Server
 - >  **esri** - Gratuit à usage non commercial
- >  **Leaflet** - <https://leafletjs.com/>
 - > OpenSource + plugin **Mapbox.js** : <https://docs.mapbox.com/help/glossary/mapbox-js/>

> Outil :

- >  **Github** - <https://github.com/>
 - > Gratuit à usage non commercial
 - > Versionning, sauvegarde des sources
 - > Partage de code et collaboration sur un projet



ArcGIS API 4.10

I. Introduction

› OpenData :

- ›  **OpenStreetMap** - <https://www.openstreetmap.fr/>
 - › Offre un service de fond de carte assez détaillé gratuit à l'échelle du monde
 - › Base de données libre - <https://www.data.gouv.fr/fr/organizations/openstreetmap/>
- ›  **GeoServer** - <http://geoserver.org/>
 - › Serveur cartographique OpenSource
 - › Publication de données : services WMS, WFS, WMTS, etc...
- ›  **GéoPortail** - <http://professionnels.ign.fr/>
 - › API et catalogue de données et services
 - › Inscrivez-vous !
- ›  **ArcGIS hub** - <https://hub.arcgis.com/pages/open-data>
 - › Données libres – services REST ArcGIS



II. Préparer son environnement de développement

Visual Studio Code

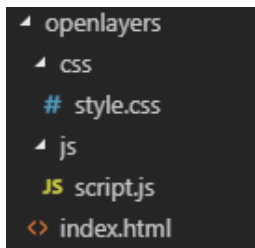


1. Installer « Visual Studio Code » - <https://code.visualstudio.com/>

- › Editeur de code (léger et gratuit fournit par Microsoft)
- › Disponible sur Windows / Mac / Linux
- › Conçu pour le web (javascript nativement supporté), peut être extensible à d'autres langages de programmation (Python, C/C++, C#) en installant des extensions
- › Possibilité d'installer de nombreuses extensions via la marketplace :
 - › ESLint, Debugger for chrome, GitHub intégration, etc...

2. Workspace

- › Choisir son espace de travail, ex : « C:\vscode-workspace\formation_javascript-2019 »
- › Créer un nouveau dossier « openlayers » qui sera la racine du site web puis créer l'arborescence suivante :



II. Débuter sur OpenLayers

Nouvelle carte ol.Map



1. Se rendre sur le site d'OpenLayers - <https://openlayers.org/>

› Go QuickStart!

- › Copier le code HTML dans votre page « index.html » - Sauvegarder ! **CTRL + S** puis ouvrir la page web dans un navigateur

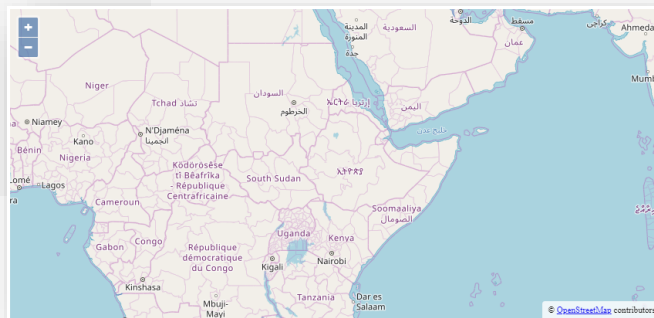
```
<script type="text/javascript">
var map = new ol.Map({
  target: 'map',
  layers: [ new ol.layer.Tile({ source: new ol.source.OSM() }) ],
  view: new ol.View({ center: ol.proj.fromLonLat([37.41, 8.82]), zoom: 4 })
});
</script>
```

› Parcourir l'API et l'objet **ol.Map**

- › <https://openlayers.org/en/latest/apidoc/>

› Ajouter l'évènement **click** sur la carte

```
map.on('click',function(event){alert('click')});
```



II. Débuter sur OpenLayers

Nouvelle carte ol.Map

2. Organiser son code

- › Déplacer le style CSS dans style.css et importer la balise `<style>` dans index.html

```
<link rel="stylesheet" href="css/style.css" type="text/css">
```

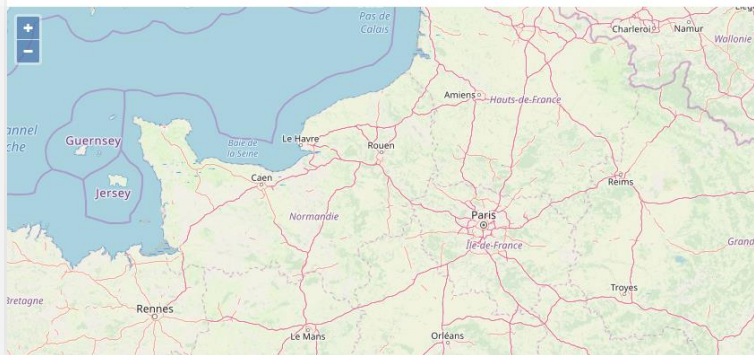
- › Déplacer le script JS dans script.js et importer la balise `<script>` dans index.html

Vérifier que ça fonctionne toujours...

```
<script src="js/script.js" type="text/javascript">
```

- › Redéfinir l'étendue initiale via la propriété `center` de l'objet `ol.View` appartenant à `Map`

Débuter sur OpenLayers



```
view: new ol.View({  
  center: ol.proj.fromLonLat([2.294481,48.858370]),  
  zoom: 6  
})
```


II. Débuter sur OpenLayers

Nouvelle carte ol.Map

3. Commenter son code !

- › La déclaration `const` indique une constante dont la valeur ne peut pas être modifiée par des réaffectations ultérieures

```
// Raster OpenStreetMap
var osm = new ol.layer.Tile({
  source: new ol.source.OSM()
});

// Map OpenLayers
const map = new ol.Map({
  target: 'map', // nom de la <div> HTML
  layers: [osm], // Couches de la carte
  view: new ol.View({ // Vue initiale
    center: ol.proj.fromLonLat([2.294481, 48.858370]), // Paris WGS84
    zoom: 6
  })
});
```

II. Débuter sur OpenLayers

Ajouter des éléments à la carte ol.control

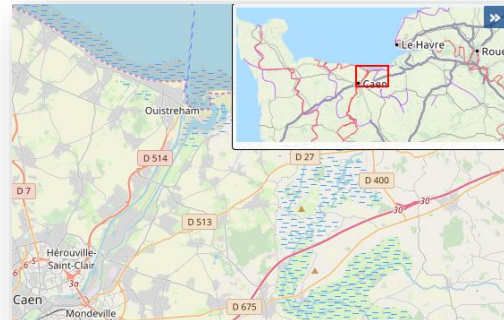
1. La propriété **controls** de l'objet **Map** définit la liste des controls attachés à la carte

- › Ajouter le control **OverviewMap** :

```
const map = new ol.Map({  
  target: 'map', // nom de la <div> HTML  
  controls: ol.control.defaults().extend([new ol.control.OverviewMap()]),  
  layers: [osm], // Couches de la carte  
  view: new ol.View({ // Vue initiale  
    center: ol.proj.fromLonLat([2.294481, 48.858370]), // Paris WGS84  
    zoom: 6  
  })  
});
```

- › La méthode **extend()** indique que l'on ajoute des controls (passés en paramètre) à ceux par défaut (Zoom, Rotate et Attribution)
- › Ajouter d'autres controls : OverviewMap, FullScreen et ScaleLine

```
controls: ol.control.defaults().extend([  
  new ol.control.OverviewMap({className: "ol-overviewmap"}), // Mini-map  
  new ol.control.FullScreen(), // Plein-ecran  
  new ol.control.ScaleLine() // Echelle  
]),
```



II. Débuter sur OpenLayers

Ajouter des éléments à la carte ol.control

2. Afficher les coordonnées X,Y

- Ajouter une `<div>` à la page HTML

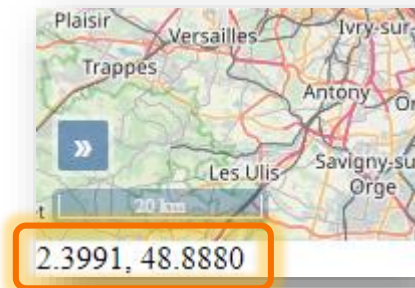
```
<div id="main">
  <div id="header">
    <h2>Débuter sur OpenLayers</h2>
  </div>
  <div id="map" class="map"></div>
  <div id="mouse-position"></div>
</div>
```

- Dans `script.js`, créer un nouveau control `ol.control.MousePosition`

```
var mousePositionControl = new ol.control.MousePosition({
  coordinateFormat: ol.coordinate.createStringXY(4),
  projection: 'EPSG:4326',
  className: 'custom-mouse-position',
  target: document.getElementById('mouse-position')
});
```

- Ajouter `mousePositionControl` à la liste des controls de la carte

```
controls: ol.control.defaults().extend([
  new ol.control.OverviewMap({className: "ol-overviewmap"}), // Mini-map
  new ol.control.FullScreen(), // Plein-ecran
  new ol.control.ScaleLine(), // Echelle
  mousePositionControl // X,Y
]),
```



II. Débuter sur OpenLayers

Ajouter des couches à la carte ol.layer

1. Ajouter une couche « WMTS »

- › <https://openlayers.org/en/latest/examples/wmts-ign.html>

```
var raster_ign = new ol.layer.Tile({
  source: new ol.source.WMTS({
    url: 'https://wxs.ign.fr/pratique/geoportail/wmts',
    layer: 'GEOGRAPHICALGRIDSYSTEMS.MAPS',
    matrixSet: 'PM',
    format: 'image/jpeg',
    projection: 'EPSG:3857',
    tileGrid: new ol.tilegrid.WMTS({
      origin: [-20037508, 20037508],
      resolutions: resolutions,
      matrixIds: matrixIds
    }),
    style: 'normal'
  })
});

map.addLayer(raster_ign); // ajout de la couche « WMTS » à la carte
```



- › <https://wxs.ign.fr/pratique/geoportail/wmts?SERVICE=WMTS&VERSION=1.0.0&REQUEST=GetCapabilities>

II. Débuter sur OpenLayers

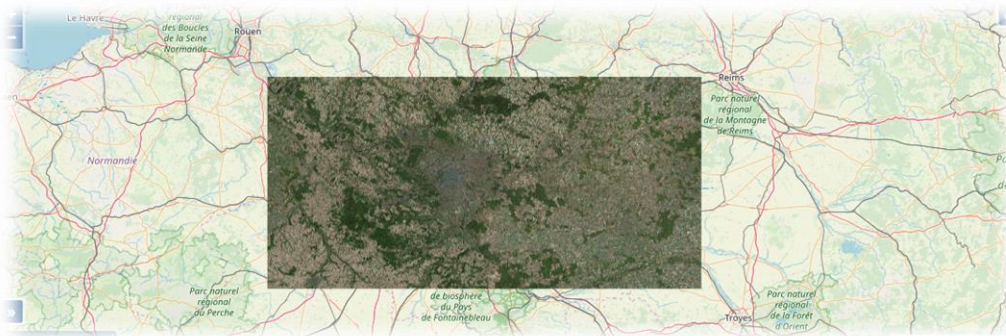
Ajouter des couches à la carte ol.layer

2. Ajouter une couche « Tiled ArcGIS MapServer »

- › <https://openlayers.org/en/latest/examples/arcgis-tiled.html?q=arcgis>

```
var ortho_esri = new ol.layer.Tile({  
  extent: [134987.79195151184,6175929.419557666,428505.98056670104,6319172.410564141],  
  source: new ol.source.TileArcGISRest({  
    url: 'https://server.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer'  
  })  
});  
  
map.addLayer(ortho_esri); // ajout de la couche « Tiled ArcGIS MapServer » à la carte
```

- › https://server.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer



II. Débuter sur OpenLayers

Ajouter des couches à la carte ol.layer

3. Visibilité des couches

- › Ajouter les « checkbox » dans la page HTML pour gérer la visibilité des couches sur la carte

```
<div>  
  <input type="checkbox" id="cbOSM" onclick="setVisibility(this)" checked><label>OpenStreetMap</label>  
  <input type="checkbox" id="cbWMTS" onclick="setVisibility(this)" ><label>WMTS</label>  
  <input type="checkbox" id="cbTileArcGISRest" onclick="setVisibility(this)" checked><label>Orthophoto</label>  
</div>
```

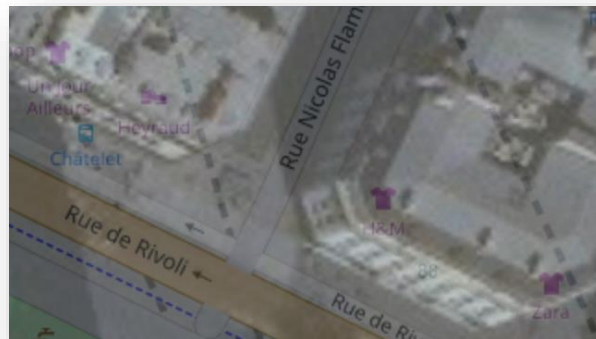
☒ OpenStreetMap ☐ WMTS ☒ Orthophoto

- › Définir la visibilité par défaut à false pour la couche WMTS

visible : false

- › Définir l'opacité à 0.7 pour l'orthophoto

opacity: 0.7



II. Débuter sur OpenLayers

Ajouter des couches à la carte ol.layer

- › Créer la fonction `setVisibility(checkbox)` dans le fichier JS

```
/**
 * Visibilité des couches
 * @param checkbox
 */
function setVisibility(checkbox) {

    map.getLayers().forEach(function(layer) {
        switch (checkbox.id) {
            case 'cbOSM':
                if (layer == osm)
                    layer.setVisible(checkbox.checked);
                break;
            case 'cbWMTS':
                if (layer == raster_ign)
                    layer.setVisible(checkbox.checked);
                break;
            case 'cbTileArcGISRest':
                if (layer == ortho_esri)
                    layer.setVisible(checkbox.checked);
                break;
        }
    });
}
```

- › Utiliser le debugger (**F12**)
 - › Ajouter un point d'arrêt et parcourir la fonction pas à pas

II. Débuter sur OpenLayers

Enrichir sa carte

1. Parcourir les exemples OpenLayers 3
 - › <https://openlayers.org/en/v3.20.1/examples/>
2. Ajouter les controls : **ZoomSlider** et **ZoomToExtent**
3. Ajouter des interactions sur la carte :
 - › Rotation
 - › Dessin et/ou mesure



II. Débuter sur OpenLayers

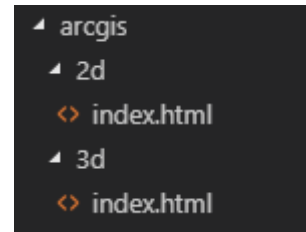
Sauvegarder ses sources sur GitHub

1. Se rendre sur GitHub - <https://github.com/>
 - › Télécharger le client « GitHub Desktop »
 - › Se créer un compte
2. Sur « GitHub Desktop »
 - › **Create a New Repository on your hard drive**
 - › Name : formation-javascript_2019
 - › Local Path : Chemin vers votre espace de travail, ex : « C:\vscode-workspace »
3. Publish repository
 - › Vérifier sur son repository en ligne la présence des fichiers
 - › Modifier un fichier et **Commit to master!**
4. Créer un Tag (version des sources à une date donnée)
 - › Release → **Create a new release**
 - › Nommer le tag : 20190315
 - › **Publish release**



III. Découverte d'ArcGIS API for JavaScript 2D / 3D

1. Dans son workspace, créer un nouveau dossier « arcgis » et 2 sous-dossiers « 2d » et « 3d » contenant une page index.html
2. Se rendre sur <https://developers.arcgis.com/javascript/>
3. Créer une nouvelle carte 2D
 - › Intro to 2D
4. Créer une nouvelle carte 3D
 - › Intro to 3D
5. Parcourir l'API
 - › <https://developers.arcgis.com/javascript/latest/api-reference/index.html>



III. Découverte d'ArcGIS API for JavaScript

Ajout de widgets

1. Parcourir les exemples de widgets :

- <https://developers.arcgis.com/javascript/latest/sample-code/index.html>

2. Ajouter le widget **BasemapGallery**

- Ajouter le **require** permettant d'accéder à la classe **esri.widgets.BasemapGallery**

```
require([
  "esri/Map",
  "esri/views/MapView",
  "esri/widgets/BasemapGallery"
], function(Map, MapView, BasemapGallery) {
```

- Créer l'objet **BasemapGallery** et l'ajouter comme composant ui (user-interface) :

```
// Widget des fonds de carte
var basemapGallery = new BasemapGallery({
  view: view2d
});

// Ajout du widget des fonds de carte à la vue 2D
view2d.ui.add(basemapGallery, {
  position: "top-right"
});
```



Imagerie



Topographie



National Geographic

III. Découverte d'ArcGIS API for JavaScript

Ajout de widgets

3. Ajouter le widget **CoordinateConversion**

- › Ajouter le **require** permettant d'accéder à la classe **esri.widgets.CoordinateConversion**
- › Créer l'objet **CoordinateConversion** et l'ajouter comme composant ui (user-interface)

```
// Widget des coordonnées
var coordinateWidget= new CoordinateConversion({
  view: view2d
});

// Ajout du widget des coordonnées
view2d.ui.add(coordinateWidget, "bottom-left");
```

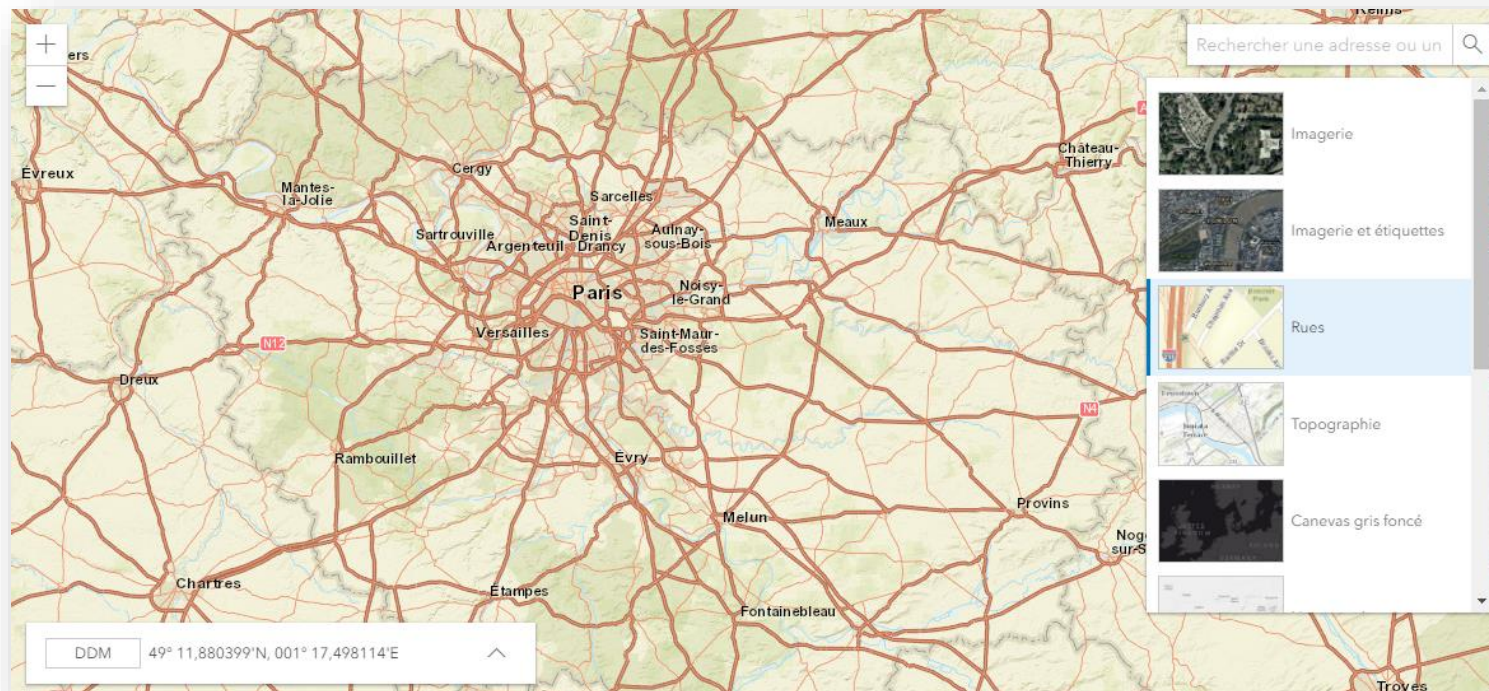
4. Ajouter le widget **SearchWidget**

```
// Widget de recherche
var searchWidget = new Search({
  view: view2d
});

// Ajout du widget de recherche
view2d.ui.add(searchWidget, {position: "top-right"});
```

III. Découverte d'ArcGIS API for JavaScript

Enrichir sa carte, ajouter d'autres widgets



› Sauvegarder ses sources sur GitHub !

Martin DELSINNE

Chef de projet en développement d'application web-sig
SIG Patrimoine
Gfi Informatique

martin.delsinne@gfi.fr

GFI Informatique
2, rue Mozart
92110 Clichy
Tél. 01 45 19 45 61
Port. 06 50 70 53 26



FRANCE | ESPAGNE | PORTUGAL | BELGIQUE | SUISSE | LUXEMBOURG |
ANGLETERRE | POLOGNE | ROUMANIE | MAROC | CÔTE D'IVOIRE |
ANGOLA | USA | MEXIQUE | COLOMBIE | BRÉSIL

gfi.world

