



JavaScript

Martin DELSINNE

Université Paris 8 - Master G2M

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

—
gfi.world

JavaScript

I. Débuter en JavaScript

Introduction

Préparer son environnement de développement Visual Studio Code

Les bases du JavaScript

II. Initiation à OpenLayers

Nouvelle carte ol.Map

Ajouter des éléments à la carte ol.control

Ajouter des couches à la carte ol.layer

Afficher une popup ol.Overlay

III. Sauvegarder ses sources

Solution de versioning GitHub

IV. Découverte de l'API ArcGIS for JavaScript v4

2D / 3D et ajout de widgets

I. Débuter en JavaScript

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**,
 - › Inutile de préciser le type des variables
 - › 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>
- › 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 » !


```
1 var myVariable;
```




I. Débuter en JavaScript

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
 - › Versioning, sauvegarde des sources
 - › Partage de code et collaboration sur un projet



I. Débuter en JavaScript

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 accès – services REST ArcGIS



I. Débuter en JavaScript

Préparer son environnement de développement - Visual Studio Code

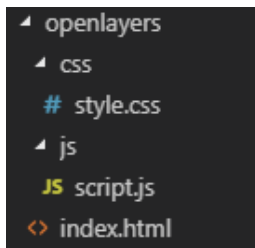


1. Installer « Visual Studio Code » - <https://code.visualstudio.com/>

- › 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 :



I. Débuter en JavaScript

Les bases du JavaScript

```
// ES6 (2015) : let pour déclarer une variable (var fonctionne toujours)
```

```
let maVariable = "Hello World"; // Affectation de la valeur
```

```
console.log(maVariable); // String
```

```
maVariable = true; // Affectation d'une nouvelle valeur 'true' de type booléen
```

```
console.log(maVariable); // Boolean
```

```
maVariable = 0; // Affectation d'un Entier
```

```
while (maVariable <= 5) { // Boucle while, avec opérateur
```

```
    console.log(maVariable); // Integer
```

```
    maVariable++; // Incréméntation
```

```
}
```

```
// Création d'un objet JS -- C'est comme un .json !
```

```
maVariable = {
```

```
    nom : "toto", // propriété 'nom', valeur '"toto"' : string
```

```
    age : 40, // propriété 'age', valeur '40' : integer
```

```
    locataire : true // propriété 'locataire', valeur 'true' : bool
```

```
}
```

```
console.log(maVariable); // Object
```

I. Débuter en JavaScript

Les bases du JavaScript

```
// Affectation d'une nouvelle propriété à l'objet  
maVariable.profession = "Apiculteur";  
console.log(maVariable); // mon objet + ma propriété
```

```
maVariable.adresse = { // Affectation d'un objet comme propriété de ma variable  
  street: "2, Rue Voltaire", // propriété 'street' : string  
  city: "44000 Nantes", // propriété 'city' : string  
  coordinate : { // propriété 'coordinate' : objet  
    x: -1.5628,  
    y: 47.2130  
  }  
}  
console.log(maVariable);
```

```
▼ {nom: "toto", age: 40, locataire: true, profession: "Apiculteur", adresse: {...}}  
  ► adresse: {street: "2, Rue Voltaire", city: "44000 Nantes", coordinate: {...}}  
    locataire: true  
    nom: "toto"  
    profession: "Apiculteur"  
  ► __proto__: Object
```


I. Débuter en JavaScript

Les bases du JavaScript – les fonctions

function()

```
/**
 * Déclaration d'une fonction nommée maFonction
 */
function maFonction() {
    // Instructions de la fonction
    console.log("Instructions à exécuter");
}
maFonction(); // Appel de ma fonction

for (let i=0;i<=5;i++) { // Boucle for
    maFonction(); // Appel de ma fonction
}

/**
 * Déclaration d'une autre fonction avec paramètres
 */
function maFonctionWithParams(param1, param2, param3) {
    // Instructions pouvant utiliser param1, param2, ...
    return "Nom : "+param1+"\nAge : "+param2+"\nProfession : "+param3; // valeur String retournée
};
// appel de ma fonction avec params qui retourne un string
console.log(maFonctionWithParams(maVariable.nom, maVariable.age, maVariable.profession));
```

I. Débuter en JavaScript

Les bases du JavaScript – les fonctions `function()`

```
// Affectation d'une fonction à l'objet --appelé aussi Method !
maVariable.getDescription = function(){
    // this indique qu'il s'agit de la propriété appartenant à l'objet stocké dans 'maVariable'
    return "Nom : "+this.nom+"\nAge : "+this.age+"\nProfession : "+this.profession; // string
};
console.log(maVariable);
console.log(maVariable.getDescription());
```

› Ajouter une balise `<p id="description"></p>` à votre page HTML

```
// Afficher la description dans <p> dont l'identifiant est 'description'
document.getElementById("description").textContent = maVariable.getDescription();
```

```
// Reinit de ma variable --null
maVariable = null;
console.log(maVariable); // null
```

I. Débuter en JavaScript

Les bases du JavaScript – les classes class

```
// Ma classe JS nommée Person
class Person {

    // 4 propriétés de l'objet initialisées par le constructeur
    constructor(nom, age, locataire, profession) {
        this.nom = nom;
        this.age = age;
        this.locataire = locataire;
        this.profession = profession;
    }

    // méthode disponible pour un objet Person
    getDescription() {
        return "Nom : "+this.nom+"\nAge : "+this.age+"\nProfession : "+this.profession;
    }

    // Instanciation d'un nouvel objet Person
    maVariable = new Person("toto", 40, true, "Apiculteur");
    console.log(maVariable);
    // var autreVariable = new Person(NOM_string, AGE_integer, LOCATAIRE_boolean, PROFESSION_string)
    // console.log(autreVariable);
```

I. Débuter en JavaScript

Les bases du JavaScript – les classes class

- › Ajouter une méthode `setAdresse(rue, ville, coord)` à la classe **Person**

```
class Person {  
  ...  
  // Autre méthode disponible pour un objet Person -- avec 3 paramètres en entrée  
  setAdresse(paramStreet, paramCity, paramCoord) {  
    // La fonction ajoute la propriété 'adresse' à l'objet instancié depuis la classe Person  
    this.adresse = { // les valeurs de l'objet sont initialisés depuis les paramètres en entrée  
      street: paramStreet,  
      city: paramCity,  
      coordinate : paramCoord  
    }  
  }  
}  
  
// Utiliser la méthode setAdresse pour initialiser la propriété 'adresse'  
// de mon objet instance de Person  
maVariable.setAdresse(  
  "2, Rue Voltaire",  
  "44000 Nantes",  
  {x: -1.5628,y: 47.2130}  
);  
console.log(maVariable);
```

I. Débuter en JavaScript

Les bases du JavaScript – les tableaux Array

> Syntaxe :

```
[element0, element1, ..., elementN]  
new Array(element0, element1[, ...[, elementN]]) dépréciée  
new Array(arrayLength) dépréciée
```

```
// Déclaration d'un tableaux [...]
```

```
var monTableau = [  
  // 1 er élément
```

```
  new Person("Pierre", 38, false, "Artisan",{  
    street:"7, rue Victor Hugo", city:"93100 Montreuil", coordinate:{x:2.439568,y:48.859413}  
  }), // virgule comme séparateur
```

```
  // 2nd élément
```

```
  new Person("Paul", 31, true, "Médecin",{  
    street:"169, rue de Bellevue", city:"92700 Colombes", coordinate:{x:2.236667,y:48.919502}  
  }) // pas de virgule car dernier
```

```
]
```

```
console.log(monTableau);
```

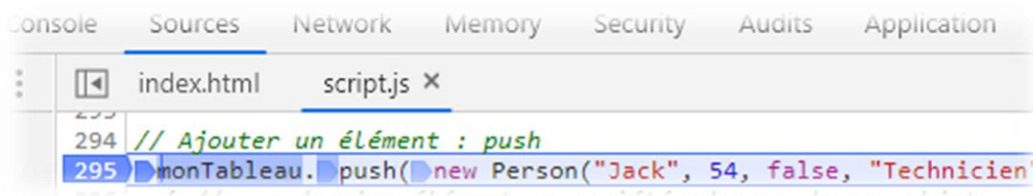
```
console.log(monTableau.length); // nombre d'élément
```

I. Débuter en JavaScript

Les bases du JavaScript – les tableaux Array

```
// Ajouter un élément : push
monTableau.push(new Person("Jack", 54, true, "Technicien",
  { // propriété adresse de mon objet
    street:"37, avenue Carnot",
    city:"94230 Cachan",
    coordinate:{
      x:2.327411,
      y:48.798287
    }
  }
));

console.log(monTableau);
console.log(monTableau.length);
```



- › Utiliser la console de debugg pour inspecter
 - › Ajouter un point d'arrêt, avancer pas à pas **F10**

I. Débuter en JavaScript

Les bases du JavaScript – les tableaux [...]

```
// Accéder à un élément
```

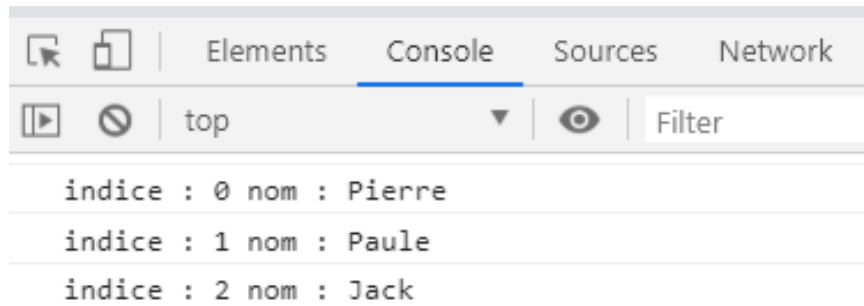
```
console.log(monTableau[1].getDescription());  
monTableau[1].nom = "Paule"; // modifier une valeur  
console.log(monTableau[1].getDescription());
```

```
/** myArray.shift(); // Retire le premier élément */  
/** myArray.pop(); // Retire le dernier élément */
```

https://developer.mozilla.org/fr/docs/Web/JavaScript/Reference/Objets_globaux/Array

```
// Parcourir le tableau, boucle for
```

```
for (var i=0;i<monTableau.length;i++) {  
  console.log( "indice : "+i.toString()+" nom : "+monTableau[i].nom);  
}
```



I. Débuter en JavaScript

Les bases du JavaScript – Les conditions if ... else ...

- › Les conditions permettent d'obtenir un résultat **vrai** ou **faux** : booléen

```
if (condition) {  
    // instructions exécutées quand la condition est vraie  
}  
else {  
    // instructions exécutées quand la condition est fausse  
}
```

```
// Parcourir les éléments du tableau  
for (var i=0;i<monTableau.length;i++) {  
    if (monTableau[i].locataire == true && monTableau[i].age < 40) { // ET Logique  
        console.log("locataire de moins de 40 ans : \n" + monTableau[i].getDescription());  
    } else if (monTableau[i].locataire == true) {  
        console.log("locataire de plus de 40 ans : \n" + monTableau[i].getDescription());  
    } else {  
        console.log("propriétaire : \n" + monTableau[i].getDescription());  
    }  
}
```


I. Débuter en JavaScript

Les bases du JavaScript – RAPPEL !!!

- › Mots clés / déclarations :
 - › **let**(es6) ou **var** : pour déclarer une variable
 - › **function** nomDeMafunction() { *//instructions de la fonction* } : pour déclarer une fonction
 - › **let** monObjet = { *//propriétés et méthodes de l'objet* } : pour créer un nouvel objet
 - › **let** monObjet = **new** nomDeMaClasse() : pour instancier un objet provenant d'une classe
 - › **Let** monTableau = [*//éléments de mon tableau séparés d'une ','*] : pour déclarer un tableau
- › Les objets instanciés contiennent 3 choses :
 - › Un **constructeur** : initialise l'état initial de l'objet (**new**)
 - › Des **propriétés** : clé / valeur
 - › Des **méthodes** : fonctions associées à l'objet
- › Les événements permettent de déclencher une fonction selon une action

II. Initiation à 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

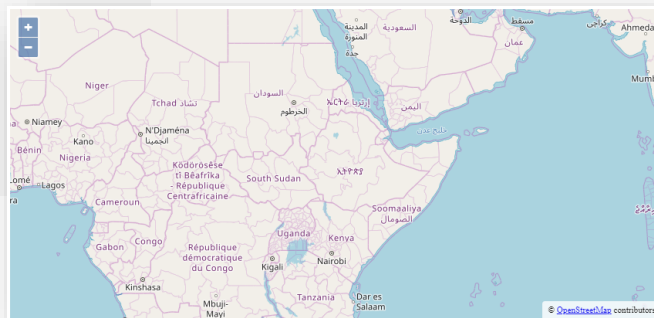
```
<script type="text/javascript">
var map = new ol.Map({
  target: 'map',
  layers: [ new ol.layer.Tile({ source: new ol.source.OSM() }) ],
  view: new ol.View({ center: ol.proj.fromLonLat([37.41, 8.82]), zoom: 4 })
});
</script>
```

- › 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. Initiation à 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">
```

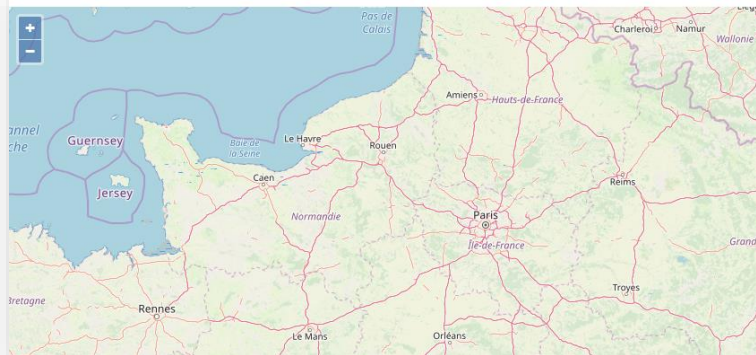
- › Déplacer le script JS dans script.js et importer la balise `<script>` dans index.html

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. Initiation à 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. Initiation à 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. Initiation à OpenLayers

Ajouter des éléments à la carte ol.control

2. Afficher les coordonnées X,Y

- Ajouter une `<div>` à la page HTML

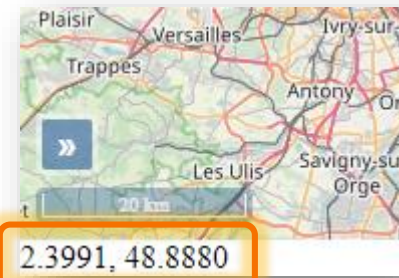
```
<div id="main">
  <div id="header">
    <h2>Débuter sur OpenLayers</h2>
  </div>
  <div id="map" class="map"></div>
  <div id="mouse-position"></div>
</div>
```

- 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. Initiation à OpenLayers

Ajouter une couche raster ol.layer.Tile

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. Initiation à OpenLayers

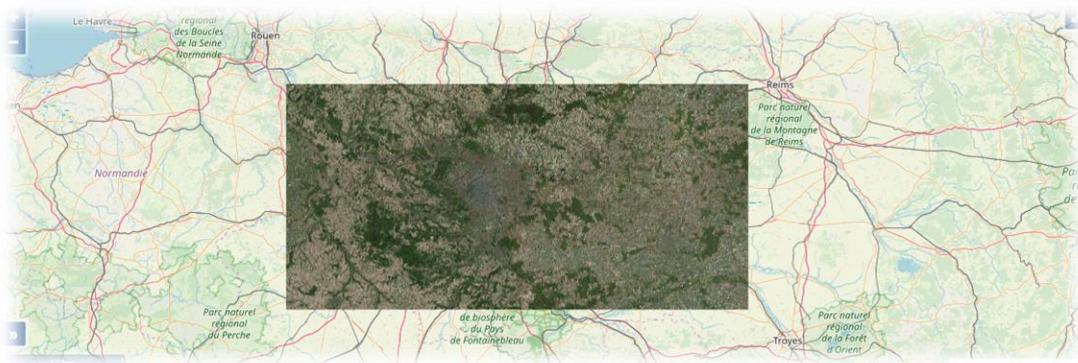
Ajouter une couche raster ol.layer.Tile

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. Initiation à OpenLayers

Ajouter une couche raster ol.layer.Tile

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
 - › Propriété **visible** de l'objet ol.source.WMTS

visible : false

- › Définir l'opacité à 0.7 pour l'orthophoto
 - › Propriété **opacity** de l'objet ol.source.TileArcGISRest

opacity: 0.7



II. Initiation à OpenLayers

Ajouter une couche raster ol.layer.Tile

- › 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
 - › Inspecter l'objet **layer** dans la boucle **forEach**



- › Ajouter une « checkbox » pour la couche vecteur

II. Initiation à OpenLayers

Ajouter une couche vecteur ol.layer.Vector

1. Créer un point

```
// Création de mon entité, c'est un objet de type ol.Feature initialisé
// avec une geom et une description
var feature = new ol.Feature({
  geometry: new ol.geom.Point(maVariable.adresse.coordinate),
  description : maVariable.getDescription()
});

/**
 * Création d'un style 'Icon'
 * @type {module:ol/style/Icon~Options}
 */
var iconStyle = new ol.style.Style({
  image: new ol.style.Icon({
    src: 'data/icon.png' // mon icône
  })
});
feature.setStyle(iconStyle); // Affectation du style à l'entité point
```

II. Initiation à OpenLayers

Ajouter une couche vecteur ol.layer.Vector

2. Ajouter une couche vecteur à la carte

```
// Création d'une couche vecteur
var vectorLayer = new ol.layer.Vector({
  source: new ol.source.Vector({ // Source de ma couche vecteur
    features: [feature] // tableau d'entités
  })
});
map.addLayer(vectorLayer); // Ajouter la couche vecteur à la carte -- method 'addLayer()'
map.getView().setCenter(maVariable.adresse.coordinate); // Centrer
map.getView().setZoom(9); // Zoom
```



II. Initiation à OpenLayers

Ajouter une couche vecteur ol.layer.Vector

3. Ajouter des entités à la couche

```
// Ajouter des entités à la couche vecteur
monTableau.forEach(function(element) { // <Array>.forEach (ES6) : parcours du tableau

    // Créer une nouvelle entité pour chaque élément du tableau
    var feature = new ol.Feature({
        geometry: new ol.geom.Point(ol.proj.fromLonLat(
            [element.adresse.coordinate.x,element.adresse.coordinate.y]
        )),
        description : element.getDescription()
    })

    // Appliquer le style
    feature.setStyle(iconStyle);

    // Ajouter à la source de la couche
    vectorLayer.getSource().addFeature(feature);

})
```

II. Initiation à OpenLayers

Afficher une popup ol.Overlay

1. Ajouter un bloc HTML

```
<div id="popup" class="ol-popup"> <!-- popup -->
  <a href="#" id="popup-closer" class="ol-popup-closer" onclick="closePopup()"></a> <!-- X fermer -->
  <div id="popup-content"></div> <!-- Contenu de la popup-->
</div>
```

2. Créer l'objet JavaScript et l'ajouter à la carte

```
/**
 * Création d'un objet ol.Overlay pour ancrer la popup sur la carte.
 */
var popup = new ol.Overlay({
  element: document.getElementById('popup') // <div>
});
map.addOverlay(popup);
```

3. Créer la fonction pour fermer la popup au clic sur X

```
// Fermer la popup
function closePopup() {
  popup.setPosition(undefined); // masquer la popup
}
```

II. Initiation à OpenLayers

Afficher une popup ol.Overlay

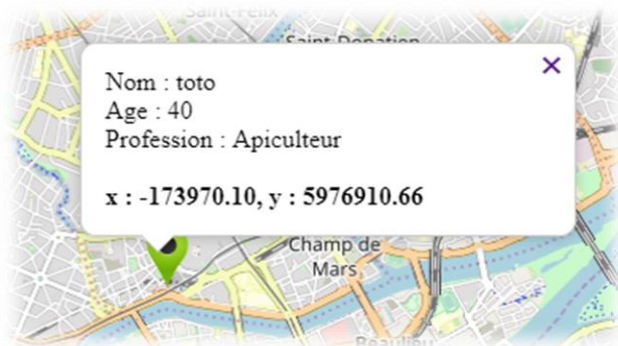
4. Modifier l'événement clic sur la carte

```
// Mon évènement click sur la carte
map.on('click', function(event){
  // Récupérer l'entité ol.Feature
  var feature = map.forEachFeatureAtPixel(event.pixel,
    function(feature) {
      return feature;
    });
  if (feature) { // Si une entité retournée
    var coordinates = feature.getGeometry().getCoordinates();
    popup.setPosition(coordinates); // Position de la popup
    //Contenu de la popup
    document.getElementById('popup-content').innerHTML =
      "<p style='margin:0'>" + feature.get('description').replace(/\n/g, '<br>') + "</p>";
  } else {
    if (document.getElementById('popup') != null) { // popup ouverte
      closePopup();
    }
  }
});
```

II. Initiation à 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 de nouvelles interactions :
 - › Rotation
 - › Dessin et/ou mesure
4. Afficher les coordonnées x,y dans la popup :



III. Sauvegarder ses sources

Utiliser une solution de versioning 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 : *date_du_jour*
 - › **Publish release**



IV. 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>

```
└─ arcgis
  └─ 2d
    <> index.html
  └─ 3d
    <> index.html
```



IV. 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"
});
```

IV. 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"});
```

IV. Découverte d'ArcGIS API for JavaScript

Enrichir sa carte

1. Parcourir les exemples de l'API v4
 - › <https://developers.arcgis.com/javascript/latest/sample-code/index.html/>
2. Ajouter la couche orthophoto

```
var orthoLayer = new TileLayer({  
  url: "https://server.arcgisonline.com/arcgis/rest/services/World_Imagery/MapServer",  
  opacity: 0.7  
});  
  
var map = new Map({  
  basemap: "streets",  
  layers: [orthoLayer]  
});
```

3. Ajouter le widget de gestion des couches
4. Ajouter le widget d'impression, imprimer ! (2D uniquement !)

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

