Programmation sous SIG

API Javascript for ArcGIS

Objectifs :

— appendre les bases de l'API Javascript for ArcGIS
— réaliser des applications web SIG à l'aide de cette API

Préalables

Dans ce TD, nous utiliserons la version 4.4 de l'API.

Les développements que vous réaliserez doivent être déposés sur un serveur web. Vous pouvez travailler en local en installant un serveur web sur votre poste (installer WAMP ou EasyPHP sous windows).

Certaines parties utilisent des données publiées sur un serveur ArcGIS. Celui de l'école est `zebra.ensg.eu`.

Ce que vous devez savoir en Javascript

Le JavaScript est un langage interprété à typage dynamique faible. Il supporte de nombreux paradigmes de programmation : orienté objets, impératif, etc. Enfin, il n'a aucun rapport avec Java !

En règle général, le code JavaScript est utilisé dans le contexte des applications web où il est exécuté dans les navigateurs.

Pour insérer du code JavaScript dans une page web, nous disposons de deux méthodes :
- l'écrire dans une balise `<script>` n'importe où dans la page HTML ;
- l'écrire dans un fichier séparé qui sera lié à la page HTML grâce à une balise `<script>` avec un attribut `src`.

Généralement nous utiliserons la seconde méthode et nous placerons la balise à la fin du `<body>`.

Fichier HTML :

```
<!DOCTYPE html>
<html>
<head>
    <title>Hello World!</title>
</head>
<body>
  <script type="text/javascript" src="helloworld.js"></script>
```

```
</body>
</html>
```

Fichier `helloworld.js` (qui ici ouvre une boîte de dialogue avec un message *Hello world !*) :

```
alert('Hello world!');
```

Quelques caractéristiques : instructions séparées par des points virgules, sensible à la casse

Commentaires :

```
//commentaire sur une ligne

/* Commentaires
sur plusieurs
lignes */
```

Pour afficher un message dans la console :

```
console.log(msg)
```

Déclaration et instanciation d'une variable :

```
// En deux étapes distinctes
var maVariable;
mavariable = 5;

// En une seule fois
var maVariable = 5;
```

Boucles :

```
for(var i = 0; i < 5; i++) {
  alert('Itération n' + i);
}
```

Classes, objets et constucteurs :

```
var objet = new Classe({
  param1: val1,
  param2: val2,
  ...
});
```

ou

```
var objet = new Classe();{
objet.param1 = val1;
objet.param2 = val2;
...
```

1 Premiers pas : afficher un fond de plan

Pour commencer, nous allons afficher un fond de carte dans une application web.

⇒ Créez un fichier HTML basique contenant pour l'instant uniquement les lignes suivantes[1] :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta name="viewport" content="initial-scale=1, maximum-scale=1">
    <title>First Web SIG application</title>
  </head>
  <body>
  </body>
</html>
```

Pour utiliser l'API Javascript for ArcGIS, nous avons plusieurs possibilités :

- la télécharger[2] et la référencer en local dans la page HTML, ce qui est utile si vous souhaitez faire des tests alors que vous n'avez pas accès à internet ;
- référencer l'API dans la page HTML en pointant vers le CDN (content delivery network), ce qui assure de travailler avec une version à jour de l'API ;
- utiliser Bower, un gestionnaire de paquets pour le web, pour installer une version spécifique de l'API en local, ce qui est utile dans certaines configurations.

Nous utiliserons dans ce TD la deuxième solution.

⇒ Ajoutez les deux balises suivantes dans l'entête de l'HTML pour référencer l'API Javascript d'ArcGIS :

```
<script src="https://js.arcgis.com/4.4/"></script>
<link rel="stylesheet" href="https://js.arcgis.com/4.4/esri/css/main.css">
```

Nous utilisons ici la feuille de style par défaut d'Esri, mais il est possible d'en utiliser d'autres[a], voir de la personnaliser.

a. https://developers.arcgis.com/javascript/latest/guide/styling/index.html

Il nous reste à ajouter le code permettant de charger les différents éléments de l'API.

⇒ Ajoutez une balise `<script type="text/javascript"src="script.js"></script>` juste avant la fermeture du `body` de votre page HTML, et créez le fichier JavaScript `script.js` dans le même répertoire que le HTML.

L'API JavaScript d'ArcGIS est basée sur la bibliothèque **Dojo Toolkit**, dont le principe de fonctionnement est donc ici repris. Cela nous impactera en particulier pour un point : le mécanisme permettant d'importer des modules ou classes de l'API.

La syntaxe sera la suivante :

```
require(["module1", "module2", ..., "dojo/domReady!"], function(Classe1,
    Classe2, ...) {
  /* écrire ici le code JavaScript pour afficher la carte */
});
```

1. La première balise meta indique que la suite de l'HTML sera encodée en UTF-8, tandis que la seconde permet de contrôler la mise en page sur les navigateurs mobiles.
2. https://developers.arcgis.com/downloads/apis-and-sdks?product=javascript

L'utilisation de `dojo/domReady` est équivalente à celle d'un attribut onLoad en HTML : le code ne s'exécute que lorsque le *document object model* (DOM) est complètement chargé.

Pour créer une carte avec l'API, nous utiliserons la classe `Map` du module `esri/Map`. C'est elle qui permet de gérer les différentes couches affichées ou encore le fond de plan. Pour rendre visible la carte, elle doit être référencée dans un objet faisant le lien entre la carte et la balise HTML où l'afficher. Cet objet c'est le `MapView` du module `esri/views/Mapview`.

⇒ Ajoutez la fonction `require()` avec le bons paramètres à votre JavaScript.

La déclaration des objets `Map` et `MapView` s'effectue de la manière suivante :

```
var myMap = new Map({
    basemap: "satellite"  // utilisation d'une image satellite comme fond de
        plan
});

var myView = new MapView({
  container: "viewDiv",  // affichage de la carte dans la balise viewDiv du
      html
  map: myMap
});
```

⇒ Recopiez ce code dans votre fonction `require()`.

Ces instructions spécifient entre autre que le contenu de la carte doit s'afficher dans une balise d'identifiant `viewDiv`.

⇒ Ajoutez cette balise dans le `body` de votre page HTML.

La carte contenue dans cette balise ne sera enfin visible que si vous lui imposez une taille minimale.

⇒ En référencant un fichier de style ou en ajoutant une balise `<style>`, précisez que la carte doit s'étendre sur toute l'étendue de la page (propriétés `height` et `width` à 100%).

Vous pouvez enfin aller dans un navigateur pour tester votre application !

Dans ce permier exemple, nous avons utilisé des options standards pour les objets `Map` et `MapView`, mais l'API permet de jouer sur de nombreux paramètres.
Pour la carte, nous pouvons par exemple changer le fond de plan. Parmis les options nous avons : streets, satellite, hybrid, topo, gray, terrain, osm, etc.
Pour la vue, il est possible de modifier le centre, le niveau de zoom, l'orientation, etc.
Consultez la documentation [a] pour la liste complète des propriétés sur lesquelles il est possible de jouer.

a. https://developers.arcgis.com/javascript/latest/api-reference/esri-views-MapView.html et https://developers.arcgis.com/javascript/latest/api-reference/esri-Map.html

2 Passer à une vue 3D

Les derniers versions de l'API Javascript for ArcGIS ont grandement simplifié la réalisation de scènes 3D. En effet, pour indiquer que l'on souhaite obtenir une vue 3D, il suffit d'utiliser une objet de type `SceneView` à la place de `MapView`. Et il n'y a rien de plus à faire !

⇒ En repartant de l'exemple précédent, réalisez une application web SIG affichant des images satellites sur un globe 3D.

Ici encore, des options de l'objet `SceneView` [a] peuvent être spécifiées dans le code : centre, niveau de zoom, modèle d'élévation, etc.

a. https://developers.arcgis.com/javascript/latest/api-reference/esri-views-SceneView.html

Excepté ce petit changement, l'utilisation de l'API est identique que l'on désire travailler en 2D ou en 3D. Pour la suite de ce TD, nous resterons en vue 3D.

3 Ajout d'une couche d'entités

Pour l'instant, notre application web SIG est assez basique. Pour la rendre plus utile, nous allons commencer par y ajouter une couche d'entités (objet `FeatureLayer` du module `esri/layers/FeatureLayer`).

Pour ajouter une couche, l'API propose globalement deux approches :
- intégrer la ressource en utilisant l'adresse de son protocole REST ;
- dans le cas d'une ressource publiée sur ArcGIS Online (ou Portal for ArcGIS), utiliser son identifiant sur la plateforme.

Nous utiliserons ici la deuxième solution. La création de la `FeatureLayer` s'effectuera donc en passant un objet de type `PortalItem` au constructeur de `FeatureLayer` :

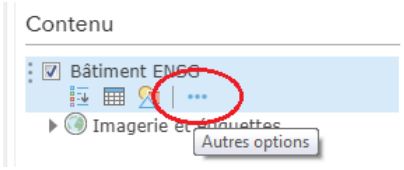
```
var featureLayer = new FeatureLayer({
  portalItem: {
    id: "<arcgis_online_feature_layer_id>"
  }
});
```

Puis une fois la couche définie, nous l'ajoutons à la carte par `myMap.add(featureLayer);`.

Nous voulons ajouter le bâtiment publié dans une carte ArcGIS Online lors du TD précédent dans notre application web SIG. Il nous faut pour cela générer préalablement la couche d'entités correspondante dans ArcGIS Online.

⇒ Sur ArcGIS Online, ouvrez la carte web réalisée lors du TD précédent (contenu *BatiENSG* de type *Web Map*).

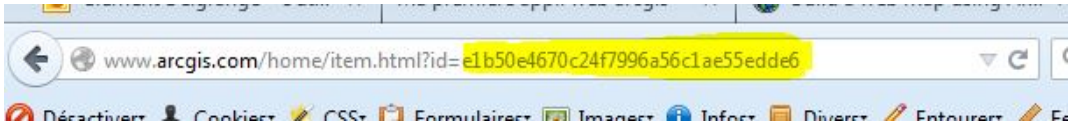
⇒ Cliquez sur les ... après la couche *Bâtiment ENSG* et sélectionnez **Enregistrer la couche**.



⇒ Revenez sur la page des contenus.

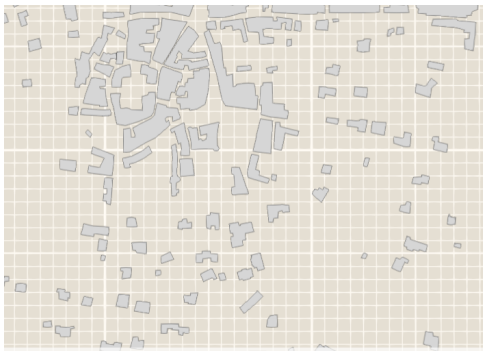
⇒ Ouvrez la ressource de type *Feature Layer*.

L'identifiant ArcGIS Online se retrouve dans l'URL :



⇒ En vous servant de la syntaxe donnée plus haut et de l'identifiant de la couche, ajoutez-la à votre application web SIG.

Depuis le début de ce TD, nous créons une carte en définissant un fond de plan. Il n'est en fait pas obligatoire d'en utiliser un. Si aucun argument n'est passé au constructeur, les entités s'affichent sur un fond blanc :



4 Déclencher une action après le chargement d'un objet

```
myView.then(function(){
  // Toutes les ressources de la vue myView ont été chargée

}, function(error){
  // Il y a eu un problème au chargement

});
```

Exemple : activer l'élévation d'une scène 3D

```
map.ground.layers.forEach(function(layer) {
```

```
  layer.visible = true;
});
```

Pour plus d'intérêt (Champs-sur-Marne c'est un peu plat), vous pouvez utiliser la couche d'entités d'identifiant "0aca29c0e8d7447282ba205f6e0942e3" sur ArcGIS Online.

afficher coordonnées clics souris

5 Bâtiments en 3D

Nous allons tenter de représenter les bâtiments en volume. Pour commencer, nous allons utiliser la même hauteur pour tous les bâtiments puis nous essayerons de tenir compte de leur hauteur réelle.

Le principe est le suivant :

- définir un `ExtrudeSymbol3DLayer`[3] en lui affectant une couleur et une taille raisonnable (10m) ;
- définir un `PolygonSymbol3D`[4] en utilisant comme couche de symboles l'`ExtrudeSymbol3DLayer` définie précédemment ;
- appliquer à la couche des bâtiments un rendu de type `SimpleRenderer`[5]) avec comme symbole celui défini juste avant.

Le principe est le même pour appliquer n'importe qu'elle symbologie à une couche : définir un symbol puis appliquer un rendu.

⇒ En vous aidant des différentes pages de la documentation, appliquez un rendu 3D aux bâtiments.

Pour faire varier la taille en fonction de la hauteur, nous allons jouer avec la propriété `visualVariables` de l'objet `SimpleRenderer`.

⇒ Regardez dans la documentation comment fonctionne cette propriété et modifiez votre code en conséquence.

6 Ajout de widgets

Les widgets sont des composants "clé-en-main" de l'API qui peuvent être utilisés dans une applications. Leurs champs d'actions sont assez variés : légende, popup, miniature, outils de mesure, de recherche ou de navigation, etc.

Nous ajouterons une légende à notre application. L'utilisation des autres widgets est similaire.

⇒ Définissez une légende à l'aide du code suivant :

```
// on crée un nouveau widget (ici une légende)
```

3. https://developers.arcgis.com/javascript/latest/api-reference/esri-symbols-ExtrudeSymbol3DLayer.html
4. https://developers.arcgis.com/javascript/latest/api-reference/esri-symbols-PolygonSymbol3D.html
5. https://developers.arcgis.com/javascript/latest/api-reference/esri-renderers-SimpleRenderer.html

```
var legend = new Legend({
  view: myView, // on indique à quelle vue s'applique le widget
  layerInfos: [{
    layer: featureLayer,
    title: "Bâtiment"
  }] // propriétés spécifiques au widget (ici : les couches et leurs noms)
});
```

⇒ Puis ajoutez la légende à la vue avec `myView.ui.add(legend, "bottom-right")`.