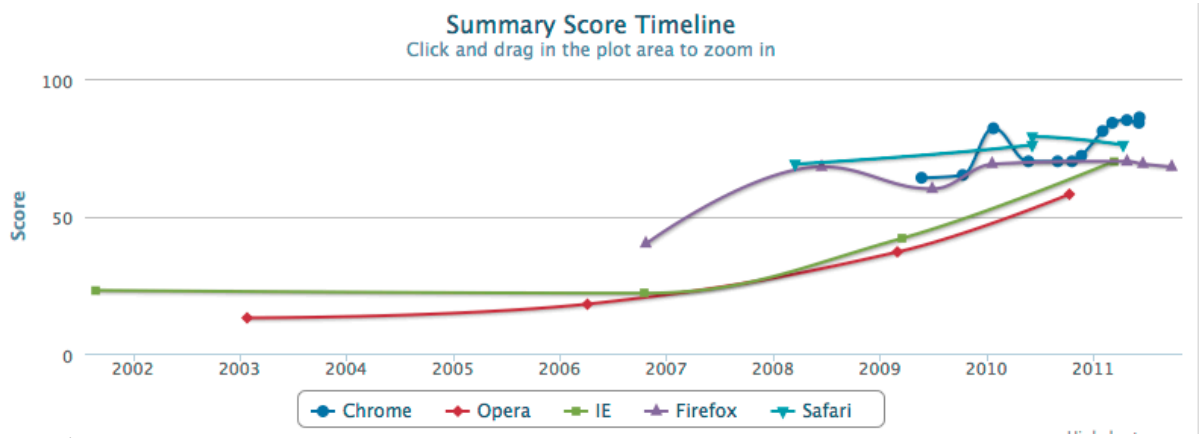
M2 Sigma – Module U904_23 décembre 2017
Laurent Jégou

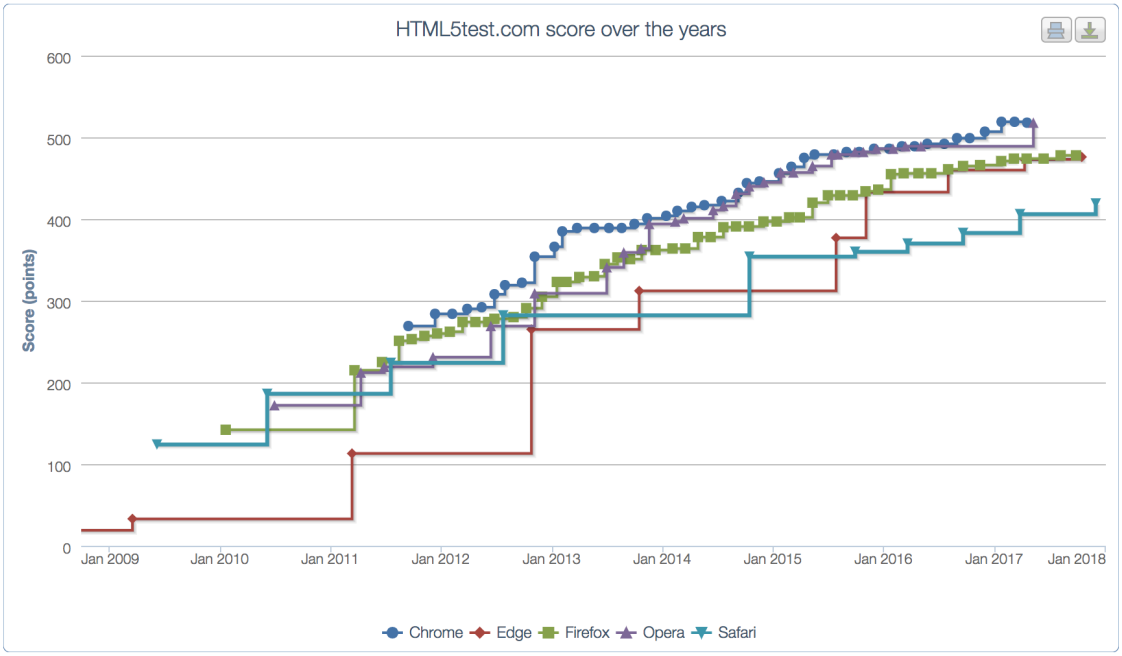
APIs et bibliothèques de fonctions JavaScript

Positionnement

L'aspect du webmapping que nous allons évoquer durant cette séance est celui qui est le plus directement visualisé et manipulé par les utilisateurs : le côté client. Cette partie des solutions de webmapping est celle qui connaît en ce moment le plus fort développement, notamment avec la progressive mise en place de grandes API[1] JavaScript et du HTML5, elle-même issue de l'amélioration corrélée de la puissance des moteurs d'exécution JavaScript des navigateurs.



Évolution du score au test "Sunspider" des principaux navigateurs depuis 2001.
Source : http ://www.browserscope.com



Score HTML5Test des principaux navigateurs de 2009 à 2017

[1] API : Application Programing Interface, ensemble de fonctions qui donnent accès à des données et des services distants. Exemple : l'API du GéoPortail., l'API Google Maps…

Outils

Le développement côté client en JavaScript ne nécessite pas d'outils logiciels très lourds, car les scripts sont exécutés (sans compilation préalable) directement par le navigateur.

Cependant il est très utile de pouvoir rédiger le code en utilisant un éditeur de texte intelligent, et ensuite de suivre son interprétation – exécution de très près par le navigateur.

Les éditeurs de texte qui sont adaptés au JavaScript sont nombreux. Parmi les plus utilisés, voici quelques références accessibles :

- Notepad ++ (windows) : http://notepad-plus-plus.org/
- Komodo Community Edition (multi) : http://www.activestate.com/komodo-edit
- NetBeans (multi) : http://netbeans.org/
- Aptana Studio : http://www.aptana.com/products/studio3.html
- Sublime Text (multi) : http://www.sublimetext.com/

Un éditeur commercial est particulièrement bien adapté au JavaScript :
https://www.jetbrains.com/webstorm/
(avec notamment un débogueur avancé).

Les débogueurs / traceurs en JavaScript les plus intéressants sont directement intégrés aux navigateurs :

- Les outils de développement de Firefox (ancviennement fireBug) : https://developer.mozilla.org/fr/docs/Outils
- Chrome Developer Tools : https://developer.chrome.com/devtools

Chrome possède l'intérêt supplémentaire de présenter un meilleur historique des appels de fonctions pour pouvoir remonter à la source d'une erreur progressivement, sur plusieurs niveaux d'appels.

Présentation

Vers 2005, plusieurs grandes sociétés du web (Microsoft, Yahoo et Google) proposent un service de cartographie accessible aux développeurs web, essentiellement comme support de services payants et de publicité. Elles ont été rejointes depuis par d'autres, comme Apple et Amazon (ou Nokia Maps devenu Here Map, puis Wego Maps) ou des opérateurs spécialisés comme Michelin (avec http://www.viamichelin.com/). Ces services consistent en des cartes (routières tout d'abord, puis des photographies aériennes et images satellite) interactives et interrogeables (avec des services comme la recherche d'adresses, le calcul d'itinéraires), cartes que l'on peut intégrer à des pages web personnelles en utilisant un accès par la programmation (API). Les services publics de cartographie, comme *l'Ordnance Survey* au Royaume-Uni et l'IGN en France (avec le GéoPortail) s'y sont mis aussi. Plus récemment, on voit se développer des offres comprenant l'hébergement (dont de services de tuiles), la réalisation de *cartes thématiques* et d'analyse, comme avec Carto (anciennement CartoDB) : https://carto.com ou MapBox : https://www.mapbox.com/.

Ces services web permettent aussi de représenter ses propres données et d'utiliser certaines fonctions d'interactivité (réaction au clic, recherches, calculs d'itinéraires). Leur fonctionnement ne nécessite aucune installation logicielle sur le serveur Internet, l'intégration des cartes et leur interactivité étant réalisée côté client, par des fonctions JavaScript organisées en API (*Application Programming Interface*, interface d'accès à une application par la programmation). **C'est donc le navigateur qui va exécuter les fonctions de cartographie, après avoir téléchargé un fichier JavaScript depuis le serveur du fournisseur de cartes ou sur le serveur http de la page.** (cf. schéma n°9 de la première séance).

1- Aperçu des possibilités de l'API Google Maps JavaScript

Documentation officielle : https://developers.google.com/maps/documentation/javascript/

Google Maps est au départ un site de visualisation de cartes et de recherches de lieux ou itinéraires. Mais la très grande quantité de données spatiales disponibles dans ce service est aussi réutilisable de manière personnalisée sur un site Internet.

La licence d'utilisation de ces services est originale : on peut utiliser librement le service dans une certaine limite d'accès et à la condition que le site Internet utilisant l'API soit accessible directement, sans identification préalable des visiteurs. Ainsi, le contenu de Google Maps n'est pas utilisable en Extra/Intranet (accès après identification) ou s'il fait partie d'un site réservé à des visiteurs abonnés. Dans ce cas il faut acquérir une licence d'utilisation de type « Premium / Entreprise » (environ 8000 euros/an). Depuis octobre 2011, l'utilisation gratuite du service de dalles cartographiques est limitée à 25 000 appels par site et par jour (Les services de géocodage et de calcul d'itinéraire, sont limités, eux, à 2 500 appels).

Il faut noter que Google se réserve la totale maîtrise du contenu cartographique diffusé par son API. Ainsi, on ne peut pas être sûr que Google n'ajoutera pas d'*information à caractère publicitaire dans son fond de carte*, comme c'est actuellement le cas sur le site de Google Maps.

Pour utiliser cette API, il faut posséder un compte Google et demander l'attribution d'une clé d'identification liée au serveur (un code d'identification individuel par projet). L'affichage de la carte est déclenché par un appel à une fonction JavaScript par le navigateur.

Cet appel comporte des paramètres pouvant préciser la taille du rectangle devant contenir la carte (typiquement un élément HTML de type DIV), les coordonnées de son point central, le niveau de zoom, et le type d'informations à afficher en fond (parmi la liste des fonds Google Maps, plan ou image satellite).

Des données personnalisées et potentiellement interactives peuvent ensuite être ajoutées à cette carte, sous la forme d'une couche qui viendrait s'y superposer, toujours en faisant appel à des fonctions JavaScript.

Préambule aux tests : procédure d'envoi de fichiers sur le serveur Sigma

Vous pouvez simplement tester la création de fichiers web, html ou PHP, sur le serveur Sigma :

- Vous connecter au serveur Pydio de la formation avec vos identifiants *sigmaetuX*

- Vous rendre dans le répertoire à votre identifiant
- Vous rendre dans le sous-répertoire « web »
- Y déposer des fichiers
- Tester leur mise en ligne par le serveur http Apache à l'adresse : http://193.55.175.126/sigmaetuX/

Cela signifie que tout ce que vous déposez dans ce répertoire se retrouve automatiquement servi sur Internet, donc attention à ce que vous y proposez.

a- Utilisation simple

Un exemple d'utilisation simple est en ligne à cette adresse :
http://193.55.175.126/commun_sigma/webmapping/google/odg.html

Comme c'est une mini-application JS côté client, il suffit d'ouvrir les outils de développement de votre navigateur pour lire le code source et interagir avec son exécution (points d'arrêt, évaluation de variables, etc.). Vous pouvez aussi copier-coller ce code source dans un nouveau fichier sur votre espace d'hébergement web du serveur Sigma.

b- Marqueurs

Une des utilisations les plus courantes de l'API Google Maps consiste à pouvoir ajouter des puces, ou marqueurs, sur une carte, pour indiquer un point d'intérêt.

Ces marqueurs peuvent être ajoutés de manière fixe dans le code, en décrivant leur position et leurs propriétés de présentation, mais on peut aussi imaginer les générer dynamiquement en allant chercher ces informations dans une base de données, par exemple pour présenter les localisations des participants d'une réunion (en utilisant un langage de script côté serveur, comme PHP).

Démonstration en ligne :
http://193.55.175.126/commun_sigma/webmapping/google/marqueur.html

Exemple avec un marqueur circulaire dimensionnable :
http://193.55.175.126/commun_sigma/webmapping/google/marqueur_cercle.html

(On peut donc dessiner des cartes thématiques en cercles proportionnels…)

Remarquez au passage que la définition d'une position dans Google Maps s'effectue avec la fonction google.maps. LatLng, avec la latitude avant la longitude (l'ordre alphabétique a prévalu sur l'ordre classique Long/Lat, X puis Y...).

Un marqueur est capable de déclencher l'événement « *click* » lorsque l'on clique dessus. Cet événement est alors traitable par le navigateur qui peut lancer une fonction JavaScript déclarée dans l'écouteur (*listener*) de cet événement.

De cette manière, on peut déclencher l'ouverture d'un objet infobulle au clic sur un marqueur.

c- Géocodage

L'un des outils les plus intéressants de l'API Google Maps JS est son géocodeur automatique, qui répond rapidement et avec une qualité inégalée parmi les outils gratuits (mais, attention, son usage est limité en volume).

Le principe est assez simple : on envoie une chaîne de caractères correspondant à une adresse, et le géocodeur répond par une position géographique (ou plusieurs, si la demande est vague ou s'il y a des lieux homonymes). Cette qualité est exprimée par l'échelle géographique à laquelle les coordonnées trouvées correspondent à l'adresse fournie, de la région au bâtiment en passant par la ville, la rue, etc.

Exemple :
http://193.55.175.126/commun_sigma/webmapping/google/geocodeur.html

Observez, au passage, *les échanges de données AJAX* (asynchrones, la page se charge sans attendre la réponse du serveur) entre la page cliente et les serveurs distants de Google (avec les outils de développement, onglet *Network*).

2- Aperçu des possibilités de l'API OpenLayers 3

OpenLayers est une bibliothèque de fonctions JavaScript développée au départ par la société MetaCarta pour ses besoins propres, puis versée dans le domaine public et retenue comme projet open source par l'OSGEO. En 2014 apparaît une version 3 complètement réécrite et modernisée, compatible avec les évolutions récentes du web : HTML5 et CSS3. La version 4 est proposée depuis le début de l'année 2017, elle propose de nombreuses optimisations et une nouvelle série d'outils d'interaction (dessin, interactions variées par l'utilisateur).

Cette bibliothèque permet, côté client, d'afficher des données géospatiales (styles), de les explorer (zoom, pan, interrogations) et d'offrir à l'utilisateur le moyen d'ajouter ses informations au clic. Pour le développeur, ces données géospatiales deviennent des objets JS manipulables et interactifs (reprojection des vecteurs, styles d'affichage, réaction au clic et même outils de dessin). OpenLayers offre donc un ensemble de fonctions utiles pour visualiser des données spatiales et permettre des interactions avancées dans le cadre d'une interface graphique complète.

OpenLayers est capable de se connecter à un grand nombre de sources de données spatiales distantes, notamment les webservices aux normes OGC, mais aussi de lire des fichiers KML, GeoJSON et des fichiers images stockés sur le serveur. OpenLayers a été sélectionné par l'IGN pour son API d'accès au GéoPortail, qui en est une extension.

Exemples avancés d'utilisation (avec la participation d'étudiants de Sigma) :

- Atlas interactif de la CAF-31 : http://www.sig-cafmipy.fr/
- Observatoire photo. des paysages de la Garonne : http://opgaronne.univ-tlse2.fr/

a- Principes de l'utilisation d'OpenLayers et première carte interactive

De nombreux exemples disponibles sur le site : http://openlayers.org/en/latest/examples/

On peut tester l'utilisation d'OpenLayers sur un serveur local. Une fois téléchargée, la bibliothèque est disponible sous deux formes : un seul fichier ol.js dont le code JS est compacté, ou un fichier ol-debug.js, dont le code JS est décompacté, donc facilement lisible, ce qui est intéressant pour le débogage. Il faut aussi lier le fichier ol.css qui contient les styles adaptés à cette version.

On appelle l'un ou l'autre de ces fichiers JS selon que l'on est en mode de développement (débogage facilité) ou en mode de production (rapidité apportée par la version « compressée »). Comme il s'agit d'une bibliothèque de fonctions JavaScript, pour l'utiliser il faut ajouter un lien dans une balise <script> de votre fichier HTML, généralement dans le bloc d'en-tête <head>.

Par exemple, si on place le répertoire OpenLayers dans un sous-répertoire nommé « ol4 » du répertoire contenant le fichier HTML l'appel sera de la forme suivante :

```
<!DOCTYPE html>
  <head>
    <meta http-equiv="content-type" content="text/html; charset=utf-8"/>
    <title>Exemple d'utilisation d'OpenLayers</title>
    <script src="./ol4/ol.js" type="text/javascript">
    </script>
</head>
...
```

Une fois la bibliothèque chargée, ses fonctions deviennent disponibles à la programmation en JavaScript. Par exemple, commençons par faire afficher une carte OpenLayers dans un de notre page HTML, carte qui sera issue des données du serveur de tuiles d'OpenStreetMap.

Exemple en ligne :
http://193.55.175.126/commun_sigma/webmapping/ol4/exemple1.html

Explication :

- Lors du chargement de la page, la fonction init() est appelée.
- Cette fonction contient les commandes OL pour créer la carte, c'est à dire :
- créer un objet carte OL dans le div dont l'identifiant (id) est "carte", composé de :
  - une couche de type TILE qui va consommer les tuiles OSM.
  - Centrée sur un objet point de coordonnées correspondant à la Haute-Garonne, en lat/long,
  - Convertie de coordonnées Spherical Mercator, le système utilisé par le WMS OSM (SRS[2] de Google Maps, utilisé aussi par les tuiles OSM).
- zoomer sur la carte au niveau 10

Notez que la div de la carte est dimensionnée par des styles « en ligne » (*inline*), c'est à dire dans la balise même de la div.

[2] Système de Référence Spatiale

b- Ajout de contrôles utilisateur et multiples couches WMS

Pour aller un peu plus loin et ajouter un peu d'interactivité, nous allons utiliser les fonctionnalités offertes par OpenLayers. Un grand nombre de fonctionnalités d'interaction utilisateur sont en effet disponibles en utilisant des objets OpenLayers simples.

Exemple en ligne ici :
http://193.55.175.126/commun_sigma/webmapping/ol4/exemple2.html

Pour réduire la charge serveur on définit l'option MaxExtent de l'objet map pour réduire l'extension des requêtes WMS qui sont lancées.

Les contrôles utilisateur OL présents sur cette page sont :

- OverviewMap (carte de localisation, en bas à gauche)
- ZoomSlider (curseur de zoom)
- ScaleLine (barre d'échelle)
- FullScreen : passage en plein écran
- MousePosition (affichage des coordonnées du la position de la souris)

OpenLayers ne propose plus de contrôle de gestion des couches. Il est possible de contrôler la visibilité des couches et leur opacité par la programmation, en modifiant les valeurs de leur proporiétés. La documentation officielle propose un exemple de ce type de contrôle, qui utilise la bibliothèque JQuery pour sélectionner rapidement les éléments de l'interface web : https://openlayers.org/en/latest/examples/layer-group.html

c- Ajout de marqueurs et d'objets vectoriels

Enfin, nous allons voir comment ajouter un marqueur (puce) cliquable sur une carte, statiquement (dans le code).

Exemple en ligne ici :
http://193.55.175.126/commun_sigma/webmapping/ol4/exemple3.html

Description du code source :

- On place des objets dans la page pour afficher non seulement la carte mais une icône et un libellé cliquable.
- On prévoit une couche tuilée (Stamen Wetercolor)
- On crée un objet de type *overlay*, qui va être positionné sur la carte (pos) et renvoyer au de la page, puis on y écrit du texte (contenu HTML).

3- Aperçu des possibilités de la bibliothèque Leaflet

http://leafletjs.com/

Leaflet est une bibliothèque JS de webmapping assez récente (2010) spécialement dédiée aux applications mobiles. Très légère (140Ko, à comparer aux 750Ko d'OpenLayers

2) elle est aussi utilisée dans de nombreux sites. Leaflet utilise nativement les évolutions récentes du web : HTML5, CSS3, tout en restant compatible avec les navigateurs plus anciens.

Exemple en ligne : http://193.55.175.126/commun_sigma/webmapping/leaflet/

L'API JavaScript est chargée localement à la ligne 6 :

```
<script src="leaflet/leaflet.js"></script>
```

Le script qui va afficher la carte doit être inclus dans une balise <script> et place à la fin de la page, juste avant la balise </body>, pour que les objets du DOM ait été créés avant d'y accéder par la programmation.

```
<script type="text/javascript">
    var map = L.map('carte').setView([43.6, 1.44], 10);
```

On crée un objet "map" en lui fournissant le nom de la div qui doit l'héberger ('carte'), puis on centre la carte sur un couple de coordonnées, enfin, on définit le niveau de zoom voulu.

La couche à afficher dans la carte est issue du serveur de tuiles de MapBox, qui est gratuit dans certaines limites, ce qui implique d'avoir créé un compte et d'utiliser une clé d'accès.

```
L.tileLayer('http://{s}.tiles.mapbox.com/v3/Clé/{z}/{x}/{y}.png', {
        attribution: 'Map data © OpenStreetMapcontributors,
              Imagery (c) <a href="http://mapbox.com">Mapbox</a>',
        maxZoom: 18
}).addTo(map);
```

Cet objet layer comporte un paramètre "attribution" qui contient le texte HTML indiquant les sources des données, puis un paramètre "maxzoom" pour éviter que l'utilisateur n'utilise un niveau de zoom inexistant dans les tuiles sources demandées.

Remarquez que les fonctions sont chaînées : on créer un nouvel objet et on lui applique des fonctions (ici addTo(map)) en les ajoutant à la chaîne avec un point.

Ensuite, on peut créer un objet 'Marker', le positionner et lui ajouter un popup, sur une ligne.

```
var marker = L.marker([43.6, 1.44]).addTo(map);

marker.bindPopup("<b>Coucou !</b><br>Je suis un popup.");
```

Leaflet possède de nombreuses extensions écrites par des sociétés / personnes tierces, qui permettent d'étendre grandement ses possibilités : sources de données (notamment SHP et Géoportail IGN), reprojection et calculs spatiaux, géocodage, cartes lissées, animation, calcul d'itinéraires, etc.

http://leafletjs.com/plugins.html