Tweete comme Pesquet

L'objectif de cet exercice est de créer une application permettant de « tweeter comme Thomas Pesquet », l'astronaute français actuellement présent dans la station spatiale internationale (ISS), en orbite autour de la Terre. En effet, lors du passage de la station au-dessus de lieux connus, l'astronaute poste une photo, accompagnée d'un message précisant l'endroit. Par exemple, lors du survol de la ville de Genève il y a quelques jours :

Le but de notre application est de fournir la photo et le nom du lieu précis, à un moment donné lors du survol de la Terre par la station. Il n'est pas demandé de réaliser la partie partage sur les réseaux. L'exercice se décompose en 3 phases : les bases de l'application, l'optimisation du résultat texte et le développement de notre propre webservice pour nos tests.

Le design[1] de l'application est totalement libre. À vous donc de faire les bons choix en matière d'ergonomie, de rendu graphique, d'interaction, etc. Pensez cette application pour le « grand public », elle ne s'adresse pas à des professionnels. Demandez-vous comment vous aimeriez qu'elle fonctionne. Trouvez lui un nom.

Vous serez également confronté à de nombreux bugs, prenez alors le temps de noter chacun d'eux et la solution mise en place le cas échéant.

Note : attention au copiés-collés ! Bien que ce soit un travail en groupe, cela ne signifie pas que vos fichiers doivent être identiques entre les groupes. Il est même difficile que ça puisse être le cas. Et vous devez vous douter qu'il est très facile de faire des recherches de similitudes. À bon entendeur, bon travail ! ;) Si jamais vous êtes bloqués, ou si vous avez des questions, n'hésitez pas à passer en L308 ou à nous envoyer des mails.

[1] Le design au sens large, tel qu'utilisé en anglais, que l'on pourrait traduire par conception

Premier niveau – Les bases de l'application

Tout d'abord, intégrez une simple carte (Leaflet, Google Maps, etc.), éventuellement avec le provider de tuile que vous souhaitez : OpenStreetMap, Mapbox (avec clé), etc.

Ensuite, faites un appel AJAX vers une API de localisation temps-réel de l'ISS :

- Par exemple, ISS Location Now à cette adresse : http://api.open-notify.org/iss-now.json
  - Appel AJAX en GET, retour en JSON
- Ajoutez un marker à la position récupérée (créez une icône personnalisée de votre choix)
- Ajoutez également quelque part la latitude/longitude en mode texte

Cet appel AJAX doit s'effectuer en boucle, après un léger délai (par exemple, toutes les secondes, voir `setTimeout` en JS). Et à chaque appel :

- La position du marker doit être mise à jour
- Le texte latitude/longitude également
- Une ligne entre le point précédent et le nouveau point doit se créer afin de voir le déplacement de l'ISS
  - Attention, lors du passage de longitude 180 à longitude -180 à ne pas créer de ligne qui « coupe » le monde. Vous pouvez par exemple ne pas ajouter de point, créer une nouvelle polyligne, etc.
- Ajoutez également un contrôle permettant de mettre à jour la position de la carte automatiquement. Par exemple :
  - case cochée, la carte suit la position de l'ISS
  - case non cochée, le déplacement est libre

Ensuite, créez un formulaire contenant :

- 3 boutons `radio` correspondants à des niveaux de zoom différents
  - Par exemple : « smartphone », « réflex » et « téléobjectif », respectivement niveau de zoom `7`, `10` et `13`
- Un bouton de validation « Tweet comme Pesquet »

Lors de la validation de ce formulaire :

- Empêchez le comportement par défaut (envoi des données au serveur)
- Création de la photo : pour cela, nous allons utiliser un service de carte statique, par exemple celui de Google (qui n'a pas besoin de clé) : voir cette URL pour en comprendre le fonctionnement : http://staticmapmaker.com/google/
  - Générez donc la bonne URL
    - latitude, longitude de l'ISS
    - zoom souhaité par l'utilisateur
  - Utilisez cette URL comme `src` d'une image HTML ou en image d'arrière-plan CSS
  - Il existe également l'API statique de Mapbox, intéressante car il est possible de passer une orientation différente (que vous pouvez générer aléatoirement entre 0 et 360°). Ce qui dans notre cas peut rajouter un coté véridique à notre prise de vue.

Cette API a toutefois besoin d'une clé, il faut donc s'inscrire (gratuit). La doc de l'API est ici : https://www.mapbox.com/api-documentation/#static

- Création du texte : pour cela, nous allons encore une fois utiliser un service, celui de geonames nommé `findNearbyPlaceNameJSON`, ici : http://www.geonames.org/export/web-services.html#findNearbyPlaceName
    - Générez donc la bonne URL
        - Latitude, longitude de l'ISS
        - username (vous aurez besoin d'en créer un, gratuit)
    - Faites un appel AJAX vers cette API
        - Traitez les résultats pour afficher un message du type : « Hello Paris, France »
        - Champs : `toponymName` ou `name`, et `countryName`
        - Ou affichez le texte par défaut « Hello World » quand pas de données

Second niveau – Optimisation du résultat texte

L'API `findNearbyPlaceNameJSON` utilisée précédemment n'est finalement pas si précise que cela, notamment en pleine mer (et on se rend vite compte que l'ISS est souvent au-dessus de l'eau). En cherchant un peu, on tombe sur une autre API de geonames : `extendedFindNearby`, ici : http://www.geonames.org/export/web-services.html#extendedFindNearby, qui renvoie les noms des mers le cas échéant. Malheureusement, cette API n'est pas disponible en JSON.

Nous allons donc créer notre propre API, qui sera un simple wrapper de l'API de geonames. Notre API doit donc :

- Faire une requête vers l'API de geonames (sur serveur)
- Traiter le résultat en XML et le retourner en JSON
- En fonction du langage utilisé (PHP, Node), vous aurez certainement besoin de `cURL` ou d'une autre librairie de requête HTTP. En Node, le module `http` peut être utilisé. Dans tous les cas, attention au proxy si vous testez depuis l'école !

Remplacez donc ensuite l'appel AJAX à l'API geonames par notre API locale, et traitez les différents résultats :

- Affichez le nom de l'océan survolé (champ `ocean`)
- Ou affichez le pays (champ `countryName`)
- Ou affichez le nom du toponyme survolé (champ `geoname` est un tableau)
- Ou affichez le texte par défaut « Hello World »

Troisième niveau – Créer notre propre API de localisation

Pour simplifier les tests et le développement (mais aussi pour se rendre compte plus rapidement de la trajectoire de l'ISS), nous allons créer notre propre API de localisation, très simpliste. Nous pourrons ainsi remplacer celle fournie précédemment et ainsi simuler la position en fonction du facteur de vitesse choisi. Notre service doit retourner la latitude/longitude de l'ISS, en fonction de sa vitesse et de sa trajectoire.

Voici les données simplifiées que nous prendrons en compte :

- La Terre a un rayon approximatif de 6371km

- La vitesse de l'ISS est d'environ 27600km/h
- L'altitude moyenne de l'ISS est de 400km
- L'inclinaison de l'ISS est de 51.64°

Pour le calcul des angles, on pourra se placer dans un repère 3d (X,Y,Z) :

- `X` positif vers latitude 0 / longitude 0
- `Y` positif vers latitude 0 / longitude 45
- `Z` positif vers latitude 90
- L'angle `azimuth` positif de X vers Y
- L'angle `polar` positif de X vers Z (mais toujours égal à 90°)

Pour cela, nous commencerons par simuler la position de l'ISS tournant autour de l'équateur, puis en modifiant l'orientation de la Terre nous aurons sa position « réelle » :

- Lors de la première connexion d'un utilisateur
  - On stocke l'heure (en millième de secondes ou millisecondes)
- Lors des connexions suivantes
  - On calcule le temps écoulé
  - On en déduit l'angle `azimuth` de la station (en fonction du temps qu'il lui faut pour réaliser un tour de la Terre à l'équateur et du facteur de vitesse voulu)
  - On calcule les coordonnées X, Y et Z du point (depuis les angles `azimuth` et `polar`)
  - On applique une rotation 3D autour de l'axe Y de 51.64°
  - On en déduit la latitude/longitude du point 3D

Cela fonctionne, mais on se rend compte que l'ISS passe aux mêmes endroits de tour en tour, tout simplement car nous n'avons pas pris en compte la vitesse de rotation de la Terre :

- Après la rotation autour de l'axe Y, ajoutez donc également une rotation autour de l'axe Z
  - La Terre tourne autour d'elle-même en 86400 secondes réelles
  - Calculez l'angle que cela représente en fonction du facteur de vitesse

Enfin, terminez en ajoutant 2 contrôles supplémentaires dans notre interface coté client :

- Un contrôle pour gérer la vitesse
- Un contrôle pour activer ou non ce service. Par exemple :
  - Case cochée : service en mode debug (avec notre propre API)
  - Case non cochée : service ISS réel