1. Qu'est-ce qu'une API ?

Les *API* (Application Programming Interface – iterface de programmation d'application) permettent de faire **dialoguer 2 logiciels** ou 2 ordinateurs entre eux. La communication est toujours basée sur une **architecture client/serveur.**

L'API présente **des fonctions et des règles** qui permettent **l'interaction et la communication entre différentes applications.** Ces interfaces facilitent l'intégration des applications, permettant aux développeurs de créer de puissants produits numériques. L'API assure la **médiation entre les applications via des requêtes et des réponses**. Par exemple, l'enregistrement dans l'application via le compte Twitter existant de l'utilisateur se produit via l'API Twitter que les développeurs ont intégrée à l'application.

Cas particulier des API REST

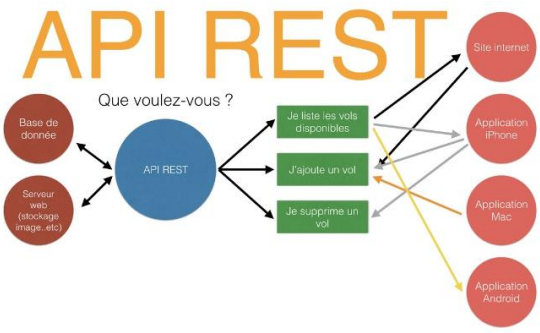
Les développeurs utilisent l'API REST (Representational State Transfer) partout où il est nécessaire de fournir des données à l'utilisateur d'une application Web ou d'un site directement à partir du serveur. C'est une API dont la conception est basée sur un style **un style architectural** via le protocole http.

Le plus souvent les développeurs utilisent des **API REST pour créer des** services web.

Comment se font les échanges avec une API ?

C'est un système de requêtes – questions/réponses - définies par le développeur.

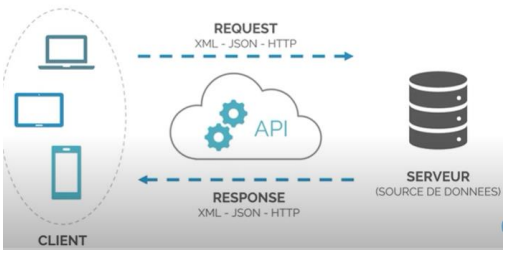
Cette liste est établie par le développeur. Elle peut être schématisée (le schéma ci-dessous se base sur une API de vols pour une agence de voyage) :



Les **requêtes** qui transitent entre le client et le serveur sont donc au **format HTTP**, auxquelles on pourra additionner des **informations/données en JSON, XML, texte**, …

La **réception** de ces données **sur le client** ne nécessite **pas de réactualisation de la page web**.

Tout ceci est schématisé ci-dessous.



Remarque : le serveur est souvent relié à une source de données.

La communication va reposer sur le protocole HTTP (compris par le client et le serveur), le serveur renvoie la ressource avec un langage que le client accepte. La réponse du serveur peut-être de différent type :

- JSON (JavaScript Object Notation), le format JSON est le plus utilisé actuellement du fait de sa simplicité.
- XML (Extensible Markup Language, comme le HTML par exemple).
- texte.
- …

Quels intérêts des API REST ?

Son principal avantage est sa **grande flexibilité**. L'architecture du service offre des **performances rapides et** une **fiabilité :**

La mise en place d'une API REST permet de **mieux cerner quelle partie de l'application a besoin de quelles données** (ou **quelle fonctionnalité back-end**). On peut alors envisager de **découper le back-end en morceaux ayant des responsabilités spécifiques** (on parle alors de services), chacun exposant une API REST.

Par exemple, un service pour la lecture / écriture des données courantes, un service pour les analyses statistiques sur les données archivées, un autre pour la génération de rapports PDF, etc.

2. La méthode *fetch* de JavaScript pour faire appel à une API

a. Qu'est-ce que la méthode *fetch* ?

fetch, nouvelle méthode créée en JS dans ES6 permet de faire appel très simplement aux API. Elle permet de créer des requêtes HTTP et de gérer les requêtes en retour sans réactualisation.

Dans les anciennes versions JS, pour faire de l'ajax on utilisait beaucoup du ***xmlhttprequest*** ou de l'ajax avec la librairie ***jquery.***

La méthode ***fetch*** simplifie beaucoup l'***ajax***.

b. Syntaxe de *fetch*

```
fetch ("http://www.site.com/xxx")    // appel de la méthode fetch. Avec une URL en
                                     paramètre pour accéder à des données.
```

Remarque : ***fetch*** récupère souvent une promesse. A traiter donc avec ***then*** et ***catch*** comme indiqué ci-dessous.

```
fetch  ("http://www.site.com/xxx")
.then  ( function(response){        // response correspond à la réponse retournée par l'API
          return response.json()    // formatage de la réponse, ici au format json (ce
                                       pourrait être au format .text pour récupérer du
                                       texte ou .blob pour récupérer des fichiers
        }
.then ( function(data){             // récupération des données cette fois que l'on pourra traiter
        console.log(data)
}) )
```

*Remarque : les fonctions en paramètres de **then** ici sont écrites de façon traditionnelle, nous avons vu à la précédente séance qu'il était possible d'écrire des fonctions fléchées. Mais également d'utiliser async/await*

Application 1

1. Ré-écrivez la syntaxe complète précédente en utilisant des fonctions fléchées en callback.
2. Complétez, comme appris à la séance précédente, une gestion des erreurs (catch).
3. Réécrivez en utilisant async/await.

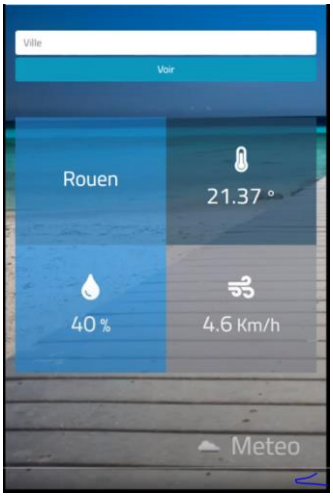
c. Application 2

Il s'agit ici d'utiliser une API créée par le gouvernement. Le but de l'application : saisir un code postal dans un formulaire afin de connaître des informations sur les villes correspondantes. Cette API entièrement publique ne nécessite pas de clé (APIkey).

4. Pour réaliser l'application suivez la vidéo suivante à parti de la 5:41 minutes : Javascript | API & Fetch - YouTube
    a. Créer un formulaire composé d'un input text pour la saisie du code postal. Un code postal peut contenir du texte, c'est le cas des codes postaux corses. Vous n'êtes pas obligé d'utiliser BootStrap comme dans la vidéo pour cela.
    b. A cette adresse, une liste d'API publics https://github.com/public-apis/public-apis
    c. Allez dans Geocoding - Utilisez l'API GeoApi
    d. Attention la page a changé depuis la création de la vidéo, Prenez le temps de retrouver les informations. Souvent savoir utiliser une API c'est savoir rechercher dans la documentation.
    e. Le but final, aller chercher via l'« APIGeoApi », les informations liées au code postal saisi telles que les noms de villes, le nombre d'habitant, le code du département … et les afficher dans une liste à puces dans la page HTML

d. Application 3

Il s'agit de créer une page web qui à partir d'une API permet d'afficher la météo d'une ville. Cette fois cette API nécessite une clé d'inscription (APIKey).
Ci-dessous l'interface de la page web à réaliser. Au chargement de la page, les informations météorologiques sont aussitôt affichées pour une ville par défaut, sur la screenshot, la ville par défaut est celle de Rouen.



1. Jouez vraiment le jeu : suivez la vidéo suivante jusqu'à la 7ième minute. A la 7ieme minute, mettez la vidéo en pause pour réaliser l'exercice en autonomie!
   Lien de la vidéo : https://www.youtube.com/watch?v=mYGwt0Vyovw
   Une fois l'exercice fini, vous pourrez reprendre la vidéo si vous le souhaitez
   a. **Créer le formulaire** avec un input « **text** » pour la ville et un input « **submit** » pour la rechercher (le submit vous permettra de voir l'utilité de *e.preventdefault()*)

   b. Parcourez la documentation de **l'API « openweather** » pour récupérer **l'url de l'APi**.
   Le site de l'API a un peu changé depuis la création de la vidéo et c'est tant mieux. Cela vous permet de rechercher dans la documentation par vous-même avec un peu d'aide.

   c. Il vous faudra **créer un compte** pour recevoir votre **clé d'api** et donc utiliser l'API.

   d. Faites-en sorte **d'afficher les informations** comme dans la copie d'écran précédente pour la **ville de Paris** par défaut.

   e. Attention, vous demanderez à l'API « OpenWeather » de vous envoyer les données en **français** et en **°c** (degrés Celsius).

   f. Complétez votre code afin qu'**au clic sur le bouton « rechercher »,** les données soient mises à jour par **appel de l'API en fonction de la ville saisie**.

   g. Le code est très ressemblant du code déjà écrit à la question d. Donc optimisez-le en **créant une ou des fonctions**.

h. Pouvez-vous récupérer et afficher l'icône **de l'API** liée à la température.

i. Moins complexe, les icônes sur la copie d'écran, sont ceux de la librairie fontawesome (il est peut-être possible d'en utiliser de chez BootStrap s'ils conviennent).