AJAX

SOMMAIRE

Edition : juin 2016 – avril 2017

AJAX

https://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript/manipulez-les-formulaires

1 - Présentation AJAX - HTTP - JSON

AJAX

http://www.w3schools.com/ajax/default.asp

Objectifs

- Ne plus avoir à faire des mises à jour complète d'une page web, mais seulement une partie
- Avoir des échanges asynchrones.

Avantages

- Diminution des temps d'attente,
- Création d'application web (ou client riche) plus proche des applications natives.

AJAX

- La programmation AJAX (Asynchronous JavaScript and XML) désigne des techniques permettant des mises à jour de page web avec des échanges serveur sans recharger toute la page.
- Ces techniques s'appuient sur JavaScript, le DOM, le protocole HTTP de requêtes asynchrones et sur sur le format JSON (remplaçant de XML).

Asynchrone

Un appel AJAX c'est une requête HTTP asynchrone.

Synchrone : j'attends la réponse, la page est bloquée jusqu'à l'arrivée de la réponse.

Asynchrone : je continue à travailler sans attendre la réponse.

Interdiction du « cross-domain »

Par défaut, une requête AJAX ne peut interroger qu'un serveur situé sur son domaine : http//monsite ne peut pas interroger http://unautresite.

Les requêtes « Cross Domain » sont interdites.

Cette limitation est faite pour des raisons de sécurité.

Un serveur web peut paramétrer son Cross-Origine Ressource Sharing pour permettre de recevoir des requêtes d'un autre site : il faut gérer ça avec prudence.

HTTP – modèle client/serveur

Rappels

HTTP: protocole client serveur.

HTTPS : équivalent sécurisé.

Client HTTP = client web = site ou appli mobile ou robot de recherche.

Serveur HTTP = serveur web = apache, etc.

Requête HTTP

https://openclassrooms.com/courses/les-requetes-http

- ***Mécanisme client-serveur***

  requête client – réponse serveur.

- ***La requête***

  Une requête HTTP, c'est un texte avec des lignes.

  Type de requête HTTP (ou méthodes HTTP)

  - GET (demande des ressources au serveur),
  - POST (envoyer des données au serveur),
  - PUT,
  - OPTION, etc.
  - Liste sur Wiki.

- ***La réponse***

  Une réponse http, c'est un texte avec des lignes.

  La réponse a un code résultat de la requête

  - 2 : succès,
  - 3 : redirection,
  - 4 : erreur du client (404 ressource non trouvée par le serveur),
  - 5 : erreur du serveur.
  - Liste sur Wiki.

JSON : format des objets JavaScript – Standard d'échange de données Web

Présentation

Le format JSON, c'est une syntaxe pour décrire de façon textuelle des informations structurées (organisées de façon précises), comme l'est le XML.

JSON : JavaScript Object Notation : c'est le format des objets JavaScript.

C'est devenu le standard actuel des échanges de données sur le Web, notamment avec AJAX. Il est supporté par de nombreux langages.

Syntaxe

Format JSON : paire « nom » / valeur (nombre, chaine, booléen, tableau, objet=structure). Tableau entre [ ] et objet entre {}

- ***Exemple :***

```
{
  "voitures" : [
    { "modèle" : "Peugeot",
      "couleur" : "bleu",
      "immatriculation" : 2008,
      "révisions" : [ 2012, 2014 ]
    },
    { "modèle" : "Citroën",
      "couleur" : "blanc",
      "immatriculation" : 1999,
      "révisions" : [ 2003, 2005, 2007, 2009, 2011, 2013 ]
    }
  ]
}
```

2 - Installation

https://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript/interrogez-un-serveur-web

Installer un serveur web local

3 étapes :

- installer le serveur Apache
- configurer appache pour autoriser les requêtes cross-domain
- créer deux fichiers de tests

1 : Installation d'Apache

Paquetage "tout-en-un" : WAMP, MAMP, XAMPP, easyPHP, etc.

Lorsqu'Apache est démarré, l'URL http://localhost correspond à la racine du répertoire de travail.

L'affichage de l'URL http://localhost/tests fonctionne sur votre machine.

Répertoire de travail d'Apache : htdocs ou www

Le répertoire de travail est un sous-répertoire du répertoire où est installé Apache. Il s'agit le plus souvent de « htdocs » sous Windows et Mac OS X, et de « www » sous Linux.

2 : Configuration d'Apache pour autoriser les requêtes cross-domain

Attention, c'est déconseillé sur un vrai serveur sans réflexion préalable !

On le fait pour le développement local sur notre machine.

Modifier le fichier de configuration : httpd.conf, en général dans un répertoire conf.

Retirer le # devant cette ligne s'il est présent

```
LoadModule headers_module modules/mod_headers.so
```

Ajouter ces lignes en fin de fichier

```
<IfModule mod_headers.c>
    # Accept cross-domain requests
    Header always set Access-Control-Allow-Origin "*"
</IfModule>
```

Sauvegarder et relancer Apache.

http://blog.inovia-conseil.fr/?p=202

3 : Création des ressources sur le serveur

Créer « sur le serveur », donc dans le répertoire de travail d'Apache, deux fichiers qu'on utilisera dans les tests :

Repertoire_de_travail / javascript-web-srv / data / langages.txt

```
[
    {
        "titre": "Ta'ang",
        "annee" : "2016",
        "realisateur": "Wang Bing"
    },
    {
        "titre": "Divines",
        "annee": "2016",
        "realisateur": "Houda Benyamina"
    },
    {
        "titre": "Juste la fin du monde",
        "annee": "2016",
        "realisateur": "Xavier Dolan"
    }
]
```

Repertoire_de_travail / javascript-web-srv / data / langages.txt

```
C++;Java;C#;PHP
```

On retrouve les deux fichiers à cette adresse : http://localhost/javascript-web-srv/data/ si Apache est bien lancé.

Le serveur est prêt à être interrogé interrogé par nos pages web.

3 - Interroger un serveur Web – AJAX – JSON

Interroger le serveur avec JavaScript - AJAX

Exemple : lire un fichier côté serveur avec JavaScript

- ***Le fichier JavaScript effectue les opérations suivantes***
  1) Création d'une requête : new XMLHttpRequest
  2) Ecriture de la requête : méthode open (GET, URL, synchrone (false))
  3) Envoi de la requête : méthode send
  4) Traitement des résultats : attribut responseText.
     - On affiche les résultats bruts en console
     - On « split » les résultats dans un tableau
     - On affiche les résultats dans la page HTML

- ***exo01 - exemple AJAX – synchrone***

- ***HTML***

  Juste un appel au JavaScript et un <ul> pour lister les résultats

```
<body>
    <h1>Quelques langages</h1>
    <ul id="langages">  </ul>
    <script src="../js/cours.js"></script>
</body>
```

- ***JavaScript***

```
// Création d'une requête HTTP
var req = new XMLHttpRequest();

// Préparation de la requête HTTP GET synchrone : false
req.open("GET", "http://localhost/javascript-web-
srv/data/langages.txt", false);

// Envoi de la requête
req.send(null);

// Traitement de la réponse reçue
console.log(req.responseText);
langages=req.responseText.split(";");
```

L'objet XMLHttpRequest

- ***Présentation***

  Tous les navigateurs ont adopté l'objet XMLHttpRequest (initialement inventé par Microsoft pour Internet Explorer).

  Il permet de créer des requêtes HTTP en JavaScript.

  http://www.w3schools.com/ajax/ajax_xmlhttprequest_create.asp

  https://developer.mozilla.org/fr/docs/Web/API/XMLHttpRequest

  https://openclassrooms.com/courses/les-requetes-http

- ***Méthode open***

  La méthode open permet de configurer la requête HTTP avant son lancement.

  3 paramètres :

  - le type de requête HTTP (le plus souvent GET, POST ou PUT),
  - l'URL cible
  - false si la requête est synchrone, true ou rien sinon.

- ***Méthode send***

  Sa méthode send envoie la requête HTTP vers l'URL cible fournie à open.

  1 paramètre :

  - L'information envoyée au serveur, quand il y en a une : requêtes POST ou PUT
  - NULL sinon : requête GET.

- ***Attribut responseText***

  L'attribut responseText contient sous forme textuelle la réponse renvoyée par le serveur à la requête HTTP.

  Il peut être splité facilement dans un tableau/

Requêtes asynchrone

Problème des requêtes synchrones

Une requête HTTP synchrone bloque la page web jusqu'à ce que la réponse soit disponible, ce qui peut prendre un certain temps.

Dans une communication avec un serveur, on prend un risque d'attente incontrôlé.

Dans l'exemple précédent, on a un message d'avertissement dans ce sens (pas dans Safari).

Passer en mode asynchrones : gestion d'événement

Si on passe en mode asynchrone, la réponse devient un événement, au même titre qu'un clic de souris.

C'est un événement « load » qui sera placé sur la requête (l'objet XMLHttpRequest).

Le traitement sera donc fait dans la méthode attachée au Listener.

Avec le mode asynchrone (pas de false dans le open), on n'a plus de message d'avertissement.

La différence entre les deux méthodes n'est perceptible qu'en cas de ralentissement du réseau.

- ***exo02 - exemple AJAX – asynchrone***

- ***JavaScript***

```
// Création d'une requête HTTP
var req = new XMLHttpRequest();

// Préparation de la requête HTTP GET asynchrone : true ou rien
req.open("GET", "http://localhost/javascript-web-
srv/data/langages.txt");


req.open("GET", //Requête GET synchrone d'un fichier du serveur
"http://localhost/javascript-web-srv/data/langages.txt");

req.addEventListener("load", function () {
    console.log(req.responseText);
    etc…
});

// Envoi de la requête
req.send(null);
```

Gestion des erreurs

Présentation

Erreurs possibles :

- URL cible incorrecte,
- serveur indisponible,
- réseau indisponible,
- etc.

Premier objectif : détecter et afficher dans la console du navigateur.

Outil

- Ecrire les erreurs en console : console.error.
- Attribut « status » et « statusText » de l'objet XMLHttpRequest. Erreur 404 : file not found !
- Evénement « error » en plus de l'événement « load »

Exemple

On va se doter d'un listener sur « load » et d'un listener sur « error ».

Selon les cas, on fait différents traitements

➢ ***exo03 - exemple AJAX – asynchrone***

➢ ***JavaScript***

Evénement load

```
req.addEventListener("load", function () {

    // Le serveur a réussi à traiter la requête
    if (req.status >= 200 && req.status < 400) {
        console.error(req.responseText);
    }
    else {
        console.error("Erreur : " + req.status + " " +
req.statusText); // info erreur
    }
});
```

Evénement « error »

```
req.addEventListener("error", function () {
    // le serveur est innaccessible
    console.error(" Erreur réseau ou serveur");
});
```

➢ ***Exemples d'erreur***

Erreur 404 : si le nom du fichier est incorrect.

Erreur réseau ou serveur : si on éteint Apache.

Généralisation de la getion d'erreur

Principes

Le code ressemble toujours à : new / open / onload / onerror / send

Dans la méthode onload, on a une partie de traitement d'erreur et une partie de traitement spécifique.

Le traitement spécifique on le met dans une méthode appelée : traitements.

On peut alors écrire une fonction générique qui reçoit une URL en paramètre ainsi que la méthode « traitements ».

On l'appellera ajaxGet()

exo04 - exemple AJAX - asynchrone avec ajaxGet

Code de ajaxGet()

Dans le open, on passe l'url

Dans le load, on place les traitements

```
function ajaxGet(url, traitements) {
  ...
  req.open("GET", url, true);

  req.addEventListener("load", function () {
    if (req.status < 200 || req.status >= 400){
      console.error("Erreur : " + req.status + " "
                    + req.statusText);
      return;
    }
    traitements(req.responseText)
  });
  ...
}
```

Utiliser la fonction générique ajaxGet

Noter qu'on travaille désormais uniquement avec la reponse qui est l'attribut responseText de l'objet XMLHttpRequest.

On crée la fonction d'affichage :

```
function afficher(reponse) {
    console.log(reponse);
}

ajaxGet("http://localhost/javascript-web-srv/data/langages.txt",
afficher);
```

ou encore, avec un fonction anonyme :

```
ajaxGet("http://localhost/javascript-web-srv/data/langages.txt",
function (reponse) {
    console.log(reponse);
});
```

Inclusion de la fonction ajaxGet (Ajax exo 3)

On fait l'inclusion de ajaxGet avant l'inclusion du traitement.

```
    <script src="../js/ajax.js"></script>
    <!-- la fonction ajaxGet doit être définie avant d'être
utilisée -->
    <script src="../js/cours.js"></script></body>
```

Parser les données JSON

Principes : JSON.parse et JSON.stringify

On a déjà vu comment transformer des données séparées par des « ; » en tableau :

Le fichier

```
C++;Java;C#;PHP
```

est récupéré dans « reponse » et on « splite » la réponse dans un tableau :

```
  langages=reponse.split(";");
```

L'objectif est maintenant de passer d'un fichier JSON à un tableau d'objets JavaScript et réciproquement. Comme JSON veut dire JavaScript Object Notation, ça devrait être facile !

2 fonctions permettent cela :

- String vers Objet : objet= JSON.parse(string)
- Objet vers String : string=JSON.stringify(objet)

Exemple 1 : travail sur un objet

Tester l'exemple dans la console de log

```
var avion = {
    marque: "Airbus",
    couleur: "A320"
};
console.log(avion);

// Transforme l'objet JSON en chaîne de caractères
var texteAvion = JSON.stringify(avion);
console.log(texteAvion);

// Transforme la chaîne de caractères en objet JSON
console.log(JSON.parse(texteAvion));
```

Exemple 2 : travail sur un tableau d'objets

Tester l'exemple dans la console de log

```
var avions = [
    {
        marque: "Airbus",
        couleur: "A320"
    },
    {
        marque: "Airbus",
        couleur: "A380"
    }
];
console.log(avions);
// Transforme le tableau d'objets JSON en chaîne de caractères
var texteAvions = JSON.stringify(avions);
console.log(texteAvions);

// Transforme la chaîne de caractères en tableaux d'objets JSON
console.log(JSON.parse(texteAvions));
```

Exercice

Afficher une liste : nom année réalisateur qui affiche des films à partir d'un fichier fims.json.

Pour cela, à partir du dernier exemple du cours, la fonction ajaxGet : chargez le fichier films.json donné au début du cours et afficher le dans la page HTML et dans la console log en suivant la même logique que dans le dernier exemple du cours pour le fichier langages.txt.

Résumé

- L'exécution de requêtes HTTP nécessite la présence d'un serveur qui publie les données nécessaires.
- Apache est à l'heure actuelle le serveur web le plus utilisé. Il doit être configuré pour accepter les requêtes HTTP sans restriction du domaine d'origine.
- L'objet JavaScript XMLHttpRequest permet de créer une requête HTTP. Sa méthode open configure la requête. Sa méthode send l'envoie vers l'URL cible.
- Une requête HTTP synchrone bloque le programme JavaScript appelant, et, par extension la page web, contrairement à une requête asynchrone qui est notifié de la réponse par le déclenchement d'un évènement load sur l'objet requête. Il est préférable d'utiliser systématiquement des requêtes asynchrones.
- Une gestion minimale des erreurs est nécessaire lorsqu'on effectue des appels AJAX. Il est possible de créer une fonction générique qui centralise le code de l'appel et la gestion des erreurs.
- Les fonctions JSON.parse et JSON.stringify permet de gérer des données JSON en JavaScript. Elles sont utiles lorsque le serveur publie des informations structurées dans ce format.

4 - Utiliser des API web

https://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript/interrogez-un-serveur-web

Introduction aux API web

Définition

Un nombre croissant de sites et de services en ligne proposent des API web destinées aux développeurs.

Une API web d'un site « S » offre aux développeurs le moyen d'exploiter les données et les services du site « S » pour leur propres site.

API (Application Programming Interface ou interface de programmation)

=

ensemble de services fournis par une application web pour d'autres applications web.

Une API va permettre :

- de récupérer la météo
- de récupérer les places libres dans un train
- de se localiser sur une carte

Plus d'info :

https://medium.com/@mercier_remi/c-est-quoi-une-api-f37ae350cb9#.4mkkuuvyd

Précisions sur la notion d'interface

Application Programming Interface

Interface : entre moi et la télé : la télécommande.

Au restaurant : le menu, interface entre moi et le serveur.

2 applications qui veulent se parler entre elles passent par une interface.

1 application propose une interface dont une autre se sert (la télévision propose sa télécommande, le restaurant propose un menu).

Une application utilise une interface proposée par une autre (j'utilise la télécommande de la télévision, j'utilise le menu du restaurant).

Par exemple, si je veux utiliser Paypal sur mon site, j'utiliserai l'API de Paypal.

Mon site peut utiliser les API d'autres sites et/ou proposer une API pour d'autres sites.

Une API déterminera quand et à quoi on peut accéder.

Contenu

- une bibliothèque de fonctions
- et/ou une bibliothèque de classes,
- et/ou des classes et des objets
- et/ou des données et des fonctions pour les utiliser
- et/ou des accès à certaines fonctionnalités qu'on peut inclure dans son programme.

Consommer et produire une API

Une application peut « consommer » des API : les utiliser. C'est l'appli-conso.

Elle peut aussi produire une ou plusieurs API pour d'autres applications. C'est l'appli-prod.

Ainsi, grâce aux API, les logiciels peuvent interagir entre eux.

Technique de base

- Une API web une API accessible via les technologies Web, notamment les protocole HTTP ou HTTPS. Elles vont souvent s'appuyer sur des technologie AJAX.
- Pour consommer (utiliser) une API web, il faut connaître son adresse (URL) et son mode de fonctionnement (souvent JSON pour les échanges de données).
- Le DOM est une API : il fournit des objets et des méthodes qui permettent à un programme JavaScript d'interagir avec une page HTML. C'est un outil générique.
- API REST : c'est un type particulier d'API. C'est un standard qui permet de normaliser le fonctionnement des API.

Exemple01 - test API 1 - fichier JSON

On utilise l'API à l'URL suivante : https://oc-jswebsrv.herokuapp.com/api/articles

Cette API, c'est juste un fichier avec un texte JSON qui est produit par une appli-prod.

On peut imaginer que ce fichier soit mis à jour régulièrement par l'appli-prod.

Ainsi l'appli-conso voit ses données elles-aussi mises à jour.

Outils

Il existe des outils pour tester les API

Navigateur

On peut copier l'URL de l'API dans le navigateur et voir ce que ça donne

Extension des navigateurs pour API – Client API

On peut installer des extensions aux navigateurs qui permettent de visualiser correctement les API ou utiliser des clients API.

Exension RestClient pour Firefox, Postman pour Chrome

Client API CocoaRestClient sur mac avec Safari.

Les API web ouvertes

Présentation

On cherche des API pour enrichir nos pages web.

Une fois l'API trouvée, il faut étudier sa documentation.

Les données sont accessibles sans clé.

- ***Exemple 1 : flickr***

  Sa documentation : https://www.flickr.com/services/api/

- ***Exemple 2 : le gouvernement***

  Exemple02 - test API 2 - premier ministre

  https://www.data.gouv.fr/fr/faq/

  https://www.data.gouv.fr/api/1/organizations/premier-ministre/

L'authentification par clé d'accès

Présentation

La plupart des API imposent des limitations d'accès.

Identification par clé d'accès (acces key)

Chaque service web peut utiliser sa propre technique pour générer ses clés d'accès, les distribuer aux clients puis surveiller leur utilisation.

La clé d'accès se présente souvent sous la forme d'une longue série de lettres et de chiffres ajoutée dans l'URL de l'API.

Ca permet au fournisseur de l'API se suivre les accès des ses clients.

Exemple : météo à Lyon

- ***Site***

  https://www.wunderground.com

- ***Weather API for developpers :***

  https://www.wunderground.com/weather/api/

  On peut demander à se créer une clé

  Ensuite, on suit la documentation :

  https://www.wunderground.com/weather/api/d/docs

- ***Exemple d'accès aux données***

  Exemple03 - exemple AJAX - test API 3 - météo Lyon

  http://api.wunderground.com/api/50a65432f17cf542/conditions/q/France/Lyon.json

iframe : API HTML intégrée

Présentation

De plus en plus de site propose une API sous la forme d'une <iframe> à intégrer directement dans le code HTML.

Dans ce cas, on n'a plus besoin de code JavaScript ou Ajax.

Exemple

Exemple04 - test API 4 - météo Paris - iframe

http://www.infoclimat.fr/api-previsions-meteo.html?id=2988507&cntry=FR

Intégration d'un code HTML avec API météo :

```
<iframe
    <iframe seamless width="888" height="336" frameborder="0"
        src="http://www.infoclimat.fr/public-
api/mixed/iframeSLIDE?_ll=48.85341,2.3488&_inc=WyJQYXJpcyIsIjQyIi
wiMjk4ODUwNyIsIkZSIl0=&_auth=BR9RRgR6UXNUeQQzAHZReFgwAzYAdgQjBnpS
MV04XiMJYgNiDm4GYFI8Ui8OIQs9AC1UNwE6BTVTOFcvWigAYQVvUT0Eb1E2VDsEY
QAvUXpYdgNiACAEIwZkUjddOF4jCW0Dbg5zBmVSPlI1DiALPgA6VDIBIQUiUzFXNF
oxAGUFYFE3BGRRNlQ%2BBGEAL1F6WG4DZAA5BG0GMFIzXTReOAlqA2EOPgZlUjRSN
g4gCzwANFQ9ATYFOlM1VzRaMgB8BXlRTAQUUS5UewQkAGVRI1h2AzYAYQRo&_c=a3
6b378f334c1f3406bd386ba10812a8"
></iframe>
```

API REST

https://openclassrooms.com/courses/utilisez-des-api-rest-dans-vos-projets-web

Présentation

REST est un protocole créé en 2000.

Une API REST suit le protocole REST.

Comme toute API, une API REST est basée sur HTTP (HyperText Transfert Protocole). Le client peut exécuter des méthodes GET, PUT, POST, DELETE, etc.

Critères d'une API REST :

- sans état (pas d'info conservé sur le client par le serveur entre deux requêtes, pas d'historique),
- avec cache (le client peut garder les informations en cache pour éviter de recharger une requête),
- orienté client serveur,
- interface uniforme,
- système de couches,
- code à la demande (optionnel).

Format des réponses du serveur pour les API REST : JSON (souvent), XML, CSV, ou même RSS.

Une API REST conforme aux standards est dite RESTful.

Alternative : SOAP

Les API SOAP sont des alternatives aux API REST.

SOAP : Simple Object Access Protocol.

SOAP définit une méthode de communication avec des règles strictes.

C'est plus sécurisé : pour les banques !

Les données sont souvent renvoyées au format XML.

Les API REST sont les plus nombreuses, largement.

Quelques exemples d'API REST

- ***Instagram API***

  Googler instagram api

  Documentation : https://www.instagram.com/developer/

  Je vais pouvoir faire de mon application un utilisateur d'instagram : elle peut récupérer tout ce qui se trouve sur instagram. Aller par exemple dans API Endpoints.

  https://api.instagram.com/v1/tags/nofilter/media/recent?access_token=ACCESS_TOKEN

  Il faudra

- ***Gmail API***

  Googler gmail api

  Avec l'api gmail, j'ai accès aux fonctionnalités d'envoi de mails. J'accède aux messages, brouillons, dossiers, etc. de l'utilisateur gmail qui sera connecté avec ses codes.

- ***Github API***

  Googler github api

  On peut héberger son code sur github. Avec l'api github, on peut récupérer les statistiques par exemple, sur mon application. On peut par exemple notifier les utilisateurs du fait qu'il se passe quelque chose.

- ***Weather API***

  Googler weather underground api

- ***Etc. !!!***

Notion de ressource

- ***Présentation***

  Une ressource c'est quelque chose que quelqu'un va chercher.

  Sur instagram, une ressource c'est une photo ou une liste de photos.

  Sur gmail, c'est un message.

  Une API organise les ressources pour qu'elles soient faciles à récupérer par les utilisateurs.

  Le format est celui de l'URI : Uniform Ressource Identifier (URL : Uniform Ressource Locator).

- ***Usage des URI***

  /users : les utilisateurs

  /users/238 : l'utilisateur 238

  /users/238/adresses : les adresses de l'utilisateur 238

  /users/238/adresses/195/notes : les notes de l'adresse 195 de l'utilisateur 238

  Cette dernière URI est trop complexe. On utilisera :

  /adresses/195/notes : cela sous-entend que c'est pour un utilisateur donné.

Résumé

- Une API est un ensemble de services offert par un logiciel à d'autres logiciels. Grâce aux API, les programmes informatiques peuvent interagir entre eux.
- Les API web sont des API accessibles via HTTP ou HTTPS. Elles utilisent souvent le format JSON. On consomme une API web dans un programme au moyen d'un appel AJAX.
- Pour consommer une API, il faut étudier sa documentation ou la tester à l'aide d'une extension de navigateur comme RESTClient ou Postman.
- Certaines API sont ouvertes. D'autres sont soumises à authentification, par exemple au moyen d'une clé d'accès.
- Une API REST est une API qui suit un protocole standardisé.

Exercices

Faire un page de synthèse de toutes les apis proposées dans le cours

https://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript/utilisez-des-api-web#/id/r-3722395

API de Github

API d'un lexique

5 - Envoyer des données à un serveur web

https://openclassrooms.com/courses/creez-des-pages-web-interactives-avec-javascript/envoyez-des-donnees-a-un-serveur-web

Configuration du serveur

Modification du fichier httpd.conf

Il faut modifier le fichier de configuration du serveur Apache.

Dans la balise <IfModule /> , il faut ajouter :

```
Header always set Access-Control-Allow-Headers "Content-Type"
```

Ce qui donne :

```
<IfModule mod_headers.c>
    # Accept cross-domain requests
    Header always set Access-Control-Allow-Origin "*"
    Header always set Access-Control-Allow-Headers "Content-Type"
</IfModule>
```

Log côté serveur

On ajoute deux fichiers php dans le répertoire MAMP/htdocs/javascript-web-srv qui vont nous permettre de suivre les envois fait au serveur dans des fichiers de log :

- ***post_form.php : va écrire dans un .log***

Ce fichier va enregistrer les données d'un formulaire qu'on trouve dans un $_FORM

```
<?php
echo "Entrée dans le fichier post_form";
if (empty($_POST)) {
    echo "Aucune donnée reçue";
}
else {
    $post = print_r($_POST, true);
    file_put_contents("post_form.log", $post);
}
```

Le code écrit $_POST dans le fichier post_form.log

Fonction file_put_contents : écrit une chaîne dans un fichier.

On retrouve le fichier à cette adresse : http://localhost/javascript-web-srv/post_form.php si Apache est bien lancé.

➢ ***post_json.php : va écrire dans un .log***

```
<?php
echo "Entrée dans le fichier post_json";
$data = print_r(json_decode(file_get_contents('php://input'),
true), true);
file_put_contents("post_json.log", $data);
```

- php://input : permet de récupérer les données d'une requêtes POST comme un fichier.
  http://php.net/manual/fr/wrappers.php.php
- file_get_contents : lit un fichier et le met dans une String
  https://www.w3schools.com/php/func_filesystem_file_get_contents.asp
- json_decode : récupère une chaîne encodée JSON et la convertit en une variable PHP.
  http://php.net/manual/fr/function.json-decode.php
- print_r(var, true) : le print_r retourne une chaine sans l'afficher (paramètre true)
  http://php.net/manual/fr/function.print-r.php

On retrouve le fichier à cette adresse : http://localhost/javascript-web-srv/post_json.php si Apache est bien lancé.

Envoyer les données au serveur : POST

Rappel de la structure d'un GET

```
var req = new XMLHttpRequest();

req.open("GET", "http://localhost/.../data/langages.txt");

req.send(null);

console.log(req.responseText);
```

Principes

L'envoi se fait via un req.open("POST",

Il existe deux techniques d'envoi :

- Intégrer les données directement dans la requête POST. C'est de cette manière que fonctionne la soumission d'un formulaire HTML.
- Transmettre les données au format JSON.

Rappels HTTP

Le POST va permettre de

- préciser l'URL qu'on veut exécuter
- passer des paramètres (dans le send)
- Le POST peut modifier, ou pas, le contenu du serveur

L'objet FormData

- L'objet FormData permet d'envoyer les données d'un formulaire via un XMLHttpRequest.
- Standardisé récemment
- Facilite l'envoi vers un serveur
- Peut être utilisé indépendamment d'un formulaire, avec sa méthode append
  https://developer.mozilla.org/fr/docs/Web/Guide/Using_FormData_Objects

exemple01 - Envoi au serveur synchrone

HTML

Juste un appel au JavaScript

```
<body>
    <h3>Qui est le plus fort ?</h3>

    <script src="../js/cours.js"></script>
</body>
```

JavaScript

Préparation d'un objet FormData : simulation de la récupération des données d'un formulaire.

```
var formData = new FormData();// Création d'un objet FormData
identite.append("login", "Bertrand"); // Ajout d'info dans
l'objet
identite.append("password", "azerty");
```

Envoi de données

```
var req = new XMLHttpRequest(); // creation d'une requête

req.open("POST",
    "http://localhost/javascript-web-srv/post_form.php");

// Envoi de la requête en y incluant l'objet
req.send(formData);
```

Analyse

1) Méthode open : on précise le fichier du serveur **qu'on veut lancer**
2) Méthode send : on passe en paramètre ce qu'on envoie. Ici, un objet FormData

Diagramme de séquence

On exécute le fichier HTML qui exécute le fichier JavaScript

On demande le traitement du fichier .php sur le serveur avec les données du formData

Fonction d'envoi générique : AjaxPost

Principes

Même logique que pour le AjaxGet

On se dote d'une fonction générique qui gère le traitement asynchrone et les erreurs de façon générique.

On l'appelle ajaxPost

Exemple

exemple02 - Envoi au serveur - Asynchrone avec ajaxPost

Envoi d'un formulaire avec FormData

FormData simplifie la récupération des données d'un formulaire par JavaScript et l'envoi AJAX.

JavaScript

```
var form = document.querySelector("form"); // noeud du formulaire

form.addEventListener("submit", function (e) { // listener

    e.preventDefault(); // arrêt de l'événement standard

    ajaxPost(
      // arg1 : l'URL du POST
      "http://localhost/javascript-web-srv/post_form.php",

      // arg2 : data : données du formulaire dans le new FormData
      new FormData(form),

      // arg3 : la fonction anonyme de traitement : ne fait rien
        function () {}
    );
});
```

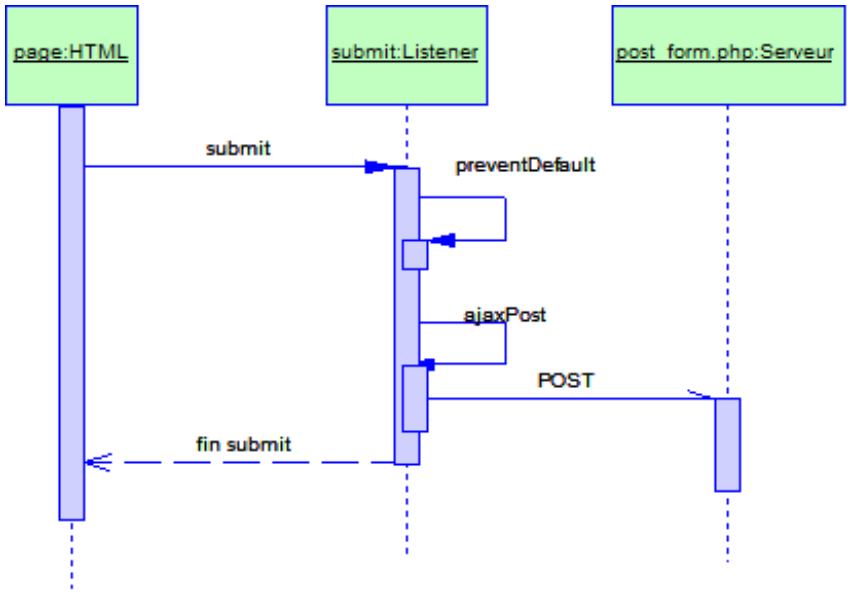
Analyse

exemple03 - Envoi au serveur - - DataForm

On suit le code par les commentaires.

On arrête l'événement standard car on ne fait pas les tests sur le serveur.

Diagramme de séquence



Quand on clique sur submit, l'événement est récupéré par le listener. Le listener a coupé la gestion d'événement standard : on passe sur la gestion d'événement submit - Ajax. On appelle ajaxPost qui envoie un POST asynchone au serveur. Fin de l'événement sumit – Ajax : la page web n'a pas été modifiée.

Envoi de données JSON

Le serveur peut aussi attendre des données au format JSON.

Dans l'exemple, on POST au serveur au chargement de la page HTML.

- ***Fonction ajaxPost avec JSON***

On ajoute le paramètre isJson à la fonction ajaxPost

```
function ajaxPost(url, data, callback, isJson) {
    … // code de la function ajaxPost
    if (isJson) {
        // Définit le contenu de la requête comme étant du JSON
        req.setRequestHeader("Content-Type", "application/json");
        // Transforme le data format JSON en data string
        data = JSON.stringify(data);
    }
    req.send(data);
}
```

A noter que les codes précédents fonctionnent encore avec la nouvelle fonction ajaxPost car JavaScript permet de ne pas passer tous les paramètres des fonctions.

- ***Fonction cours.js***

```
// Création d'un objet représentant un film
var film = {
    titre: "Zootopie",
    annee: "2016",
    realisateur: "Byron Howard et Rich Moore"
};
// Envoi de l'objet au serveur
ajaxPost("http://localhost/javascript-web-srv/post_json.php",
film,
    function (reponse) {
        // Le film est affiché dans la console en cas de succès
        console.log("Le film " + JSON.stringify(film) + " a été
envoyé au serveur");
    },
    true // Valeur du paramètre isJson
);
```

TD

1) Installer un environnement de travail : configuration d'Apache, création des fichiers. Chapitre 2 et 5.
2) Tester les exemples des chapitres 3, 4 et 5
3) Faire le TD du chapitre 3 et le TD du chapitre 4.