AJAX

SOMMAIRE

Edition : juin 2016 – avril 2017 – juin 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

Historique

Terme inventé en 2005. Le début date d'Internet Explorer 5, en 1999, avec l'introduction de l'objet XMLHttpRequest par Internet Explorer.

Objectifs

- Ne plus avoir à faire des mises à jour complète d'une page web, mais seulement une partie
- Avoir des échanges asynchrones : le client fait une demande mais n'attends pas la réponse.

AJAX

- Il s'agit d'un ensemble de méthodes intégrées dans JavaScript pour transférer, « en coulisse », des données entre le navigateur et le serveur.
- 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 le JavaScript, le DOM et le protocole HTTP de requêtes asynchrones.
- AJAX permet de demander tout type de donnée textuelle : des fichiers texte, du HTML, du XML et aujourd'hui beaucoup de format JSON. Mais aussi des nombres, des données d'un tableur (en csv), etc.

Avantages

- Diminution des temps d'attente
- Réduction de la quantité de données à envoyer et recevoir
- Création d'application web (ou client riche) plus proche des applications natives.

Exemple

Google Maps : de nouvelles sections d'une carte sont téléchargées depuis le serveur quand elles sont nécessaires sans imposer de recharger complètement la page.

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

Bases

HTTP : protocole de communication 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.

  Principales requête HTTP (ou méthodes HTTP) :

  - GET : demande des ressources au serveur. Le GET envoie de l'information pour dire ce qu'il veut et il reçoit la réponse. Un GET ne modifie pas l'état du serveur. On peut répéter un GET plusieurs fois sans qu'il y ait de conséquences sur le serveur. Par exemple pour récupérer le contenu d'un fichier. Par exemple : un download est un GET.
  - POST : envoyer des données au serveur pour qu'il en fasse quelque chose. Un POST modifie l'état du serveur. Si on fait plusieurs POST, il y plusieurs effets. Par exemple : un « j'aime » sur un réseau social est un POST.
  - Liste complète sur Wiki.

- ***La réponse***

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

  La réponse a un code résultat de la requête

  Principaux code de réponse :

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

2 - Installation

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

Installer un serveur web local

Pour pouvoir tester sur sa machine de l'AJAX, il faut installer un serveur web local : autrement dit un environnement type WAMP.

3 étapes :

- installer le serveur Apache
- configurer apache 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 / films.json

```
[
    {
        "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 – page web – XML - JSON

Présentation

On va montrer ici plusieurs techniques AJAX

Récupération d'une page web et affichage (c'est du HTML) : méthodes POST et GET

Récupération d'un fichier XML : méthode GET. Les flux RSS sont gérés en XML.

Récupération d'un fichier JSON : méthode POST

Requêtes synchrones et asynchrones

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.

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.

Le traitement sera donc fait dans la méthode attachée à un Listener..

Etapes générales du code

1) Création d'une requête HTTP : new XMLHttpRequest ()
2) Ecriture de la requête : méthode open ()
3) Envoi de la requête : méthode send ()
4) Création d'un listener : attribut onreadystatechange
5) Ecriture des traitements dans la fonction du listener (le callback)

L'objet XMLHttpRequest

Présentation

Tous les navigateurs ont adopté l'objet XMLHttpRequest.

Il a été initialement inventé par Microsoft pour Internet Explorer en 1999 comme objet ActiveX. ActiveX est une technologie de dialogue entre programme permettant, entre autres, de gérer des mises à jour de logiciel. Les navigateurs ont gardé l'objet sans prendre toute la technologie ActiveX

Il permet de créer des requêtes HTTP en JavaScript.

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

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

Les méthodes

- ***open (methode, url, asychrone)***

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.

➢ ***send (data)***

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.

➢ ***setRequestHeader (clé, valeur)***

La méthode permet de définir des couple clé-valeur (champ-valeur) pour configurer un peu plus la requête en cas de POST. Il faut définir le content-type.

Paramètres usuels pour un fichier texte : ("Content-type", "application/x-www-form-urlencoded")

Les attributs

➢ ***responseText***

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

Ca peut être du code HTML. Ou un format JSON : dans ce cas, il peut être transféré facilement dans un tableau

➢ ***responseXML***

Les données sont envoyées au format XML. Elles sont alors exploitables avec le DOM.

➢ ***readyState***

Indique le statut de la requête : 0=non initialisée, 1=chargement en cours, 2=chargée, 3=interactive, 4=terminée.

➢ ***status***

Code de retour de serveur. Utile pour la gestion des erreurs.

Les erreurs les plus courantes sont :

- URL cible incorrecte
- serveur indisponible
- réseau indisponible

Liste des codes erreurs : http://www.codeshttp.com

Attribut d'événement

➢ ***onreadystatechange***

On peut lui affecter une fonction qui sera appelée à chaque fois que le readyState change.

Exemple n°1 : récupération d'une URL HTML avec un POST

Cœur du Script JS :

```
    request = new XMLHttpRequest();
    request.open("POST", "urlpost.php")
    // couple clé-valeur pour le header : pour le POST
    request.setRequestHeader("Content-type", "application/x-www-
form-urlencoded")
    params = "url=amazon.fr/gp/aw"
    request.send(params);
```

C'est un POST :

- on a un setRequestHeader
- l'url est dans le send()

Traitements asynchrone

```
request.onreadystatechange = function() {
     console.log("this.readyState : ", this.readyState);
     // CAS D'ERREUR
     if(this.readyState != 4) return;
     if(this.status != 200){
       alert("Erreur Ajax : aucune donnée reçue")
       return;
     }
     if(this.responseText == null){
       alert("Erreur Ajax : "+ this.statusText)
       return;
     }

     // CAS GENERAL
     document.getElementById("info").innerHTML=this.responseText

     console.log("this.responseText : ")
     console.log(this.responseText);
}
```

On commence par la gestion des cas d'erreur.

Le cas général consiste à mettre la réponse dans un innerHTML

Le fichier PHP

Il fait uniquement un echo du fichier qu'il a reçu en paramètre :

```
  if (isset($_POST['url'])){
    echo file_get_contents(
      'http://' . nettoieString($_POST['url'])
    );
  }
```

Le file_get_content transforme le fichier en chaine de caractères.

La fonction nettoieString permet de faire du ménage et de gérer la sécurité.

- ***Principe : l'echo est retourné au client***

C'est l'écho que le client reçoit.

Si on avait fait echo « coucou ! » on afficherait « coucou » dans la page HTML.

Bilan

On peut récupérer des pages en asynchrone.

On pourrait faire un accès direct, mais ça risquerait de ne pas fonctionner car l'AJAX entre domaines différents ne marche pas. Ici l'AJAX est sur le même domaine (le client et le serveur) et le serveur fait l'appel synchrone hors AJAX de l'URL.

Exemple n°2 : récupération d'une URL HTML avec un GET

Cœur du Script JS :

```
request = new XMLHttpRequest();
url = "amazon.fr/gp/aw"
request.open("GET", "urlget.php?url="+ url)
request.send(null); // le paramètre est sur l'url
```

C'est un GET :

- Pas de paramètre dans le send
- Pas de setRequestHeader
- L'url est dans le open ()

Traitement asynchrone

C'est le même que précédemment

Exemple n°3 : récupération de XML avec un GET

Cœur du Script JS :

C'est le même que le précédent, sauf que l'url est un fichier XML

```
url = "www.lemonde.fr/culture/rss_full.xml"
```

Traitement asynchrone

On retrouve la même gestion d'erreurs.

Le cas général change : on peut parcourir le fichier XML avec le DOM. Si on connaît la structure, c'est facile. Sinon, on peut afficher le fichier en console.log et regarder les balises.

Ici, on trouve une balise <title> qu'on va exploiter :

```
// innerHTML  : var globale pour le callback
innerHTML = "";
request.onreadystatechange = function() {

  ...

  // CAS GENERAL
  titles=this.responseXML.getElementsByTagName("title")
  for(var i=0; i<titles.length; i++)
    innerHTML+=titles[i].textContent +'<br>'
  document.getElementById("info").innerHTML=innerHTML

  console.log("this.responseText : ")
```

```
      console.log(this.responseText);
    }
```

Intérêt du XML

On a tous les outils pour l'exploiter avec le DOM. Si on connaît l'organisation de l'arbre des données, on pourra faire tout ce qu'on veut !

Exemple n°4 : récupération de données JSON

Cœur du Script JS :

C'est le même pour un GET de fichier texte (exemple 2)

Mais l'url est en local : on récupère les données sur notre machine : adresse http://localhost/etc.

```
  request = new XMLHttpRequest();
  url = "http://localhost/javascript-web-srv/data/films.json"
  request.open("GET", url)
  request.send(null); // le paramètre est sur l'url
  url = "www.lemonde.fr/culture/rss_full.xml"
```

Traitement asynchrone

C'est la même chose que pour le XML : il faut faire un traitement des données.

```
    // CAS GENERAL
    console.log("this.responseText : ")
    console.log(this.responseText);

    document.getElementById("info").innerHTML='Liste des films';

    var films = JSON.parse(this.responseText);
    // Affiche le titre de chaque film
    films.forEach(function (film) {
           console.log(film.annee + " - " + film.titre+ " - " +
film.realisateur);
      var noeud = document.createElement("li");
      noeud.textContent = film.annee + " - " + film.titre+ " - "
+ film.realisateur;
      document.getElementById("info").appendChild(noeud); // lien
du noeud avec son parent
    });
```

On JSON.pase le responseText : ça met les données dans un tableau d'objets.

Ensuite on parcourt ce tableau pour ajouter des li dans la div.

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 ]
    }
  ]
}
```

Parser les données JSON

Principes : JSON.parse et JSON.stringify

L'objectif est 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));
```

Fonction split() : parser n'importe quelle série de valeurs séparés par un séparateur

Si on récupère le ficiher suivant :

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

On peut le transformé en tableau avec la fonction split :

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

On pourra appliquer le même principe sur un fichier csv

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 propre 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.

Sur une console Mac, je peux taper :

```
wget 'https://oc-jswebsrv.herokuapp.com/api/articles'
```

Ca va charger le fichier.

C'est un GET http.

Remarques sur le code

On utilise une fonction générale appelée : ajaxGet (url, traitements())

Cette fonction reçoit une url et une fonction en paramètre.

Elle fait la requête HTTP et la gestion des erreurs.

Ainsi, on n'a plus qu'à coder la fonction de traitement :

```
ajaxGet("https://oc-jswebsrv.herokuapp.com/api/articles",
function (reponse) {

}
```

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 () {}
    );
});
```

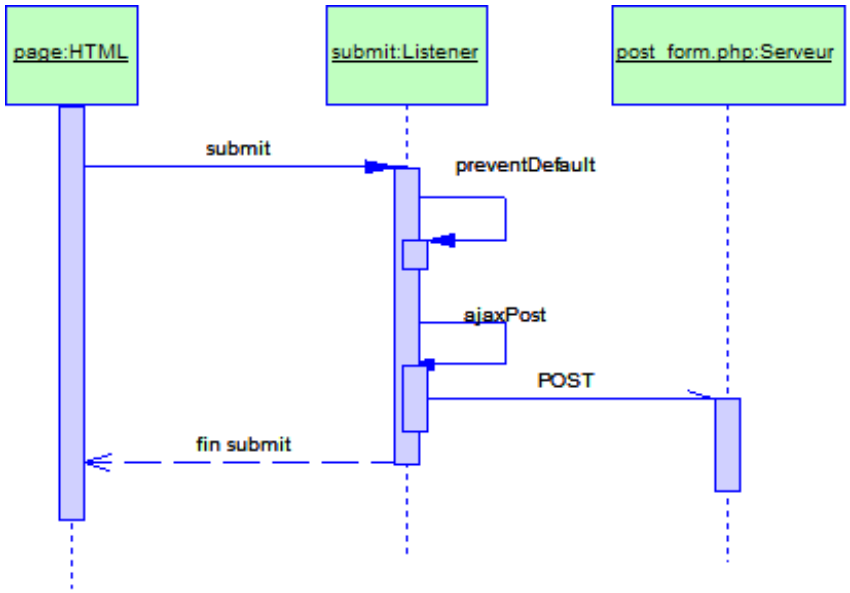
Analyse

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
);
```

Application concrète

Communiquer avec un serveur grâce à AJAX pour ne pas avoir à recharger les pages après validation d'un formulaire.

TD

Série 1

1) Installer un environnement de travail : configuration d'Apache, création des fichiers. Chapitre 2 et 5.
2) Tester l'exemple 1 qui est fourni avec le cours.
3) Coder et faire fonctionner les exemples 2, 3 et 4. Pour l'exemple 3, on cherchera à afficher d'autres informations issues du XML que celles proposées dans le cours.

Le fichier JSON est fourni avec l'exemple 1.

Série 2

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

Série 3

Organisez des données d'étudiants dans une classe avec 1 note par matière à une certaine date en format JSON et XML