LES API HTML5
API AUDIO, VIDEO

BEN DAKHLIA Sonia

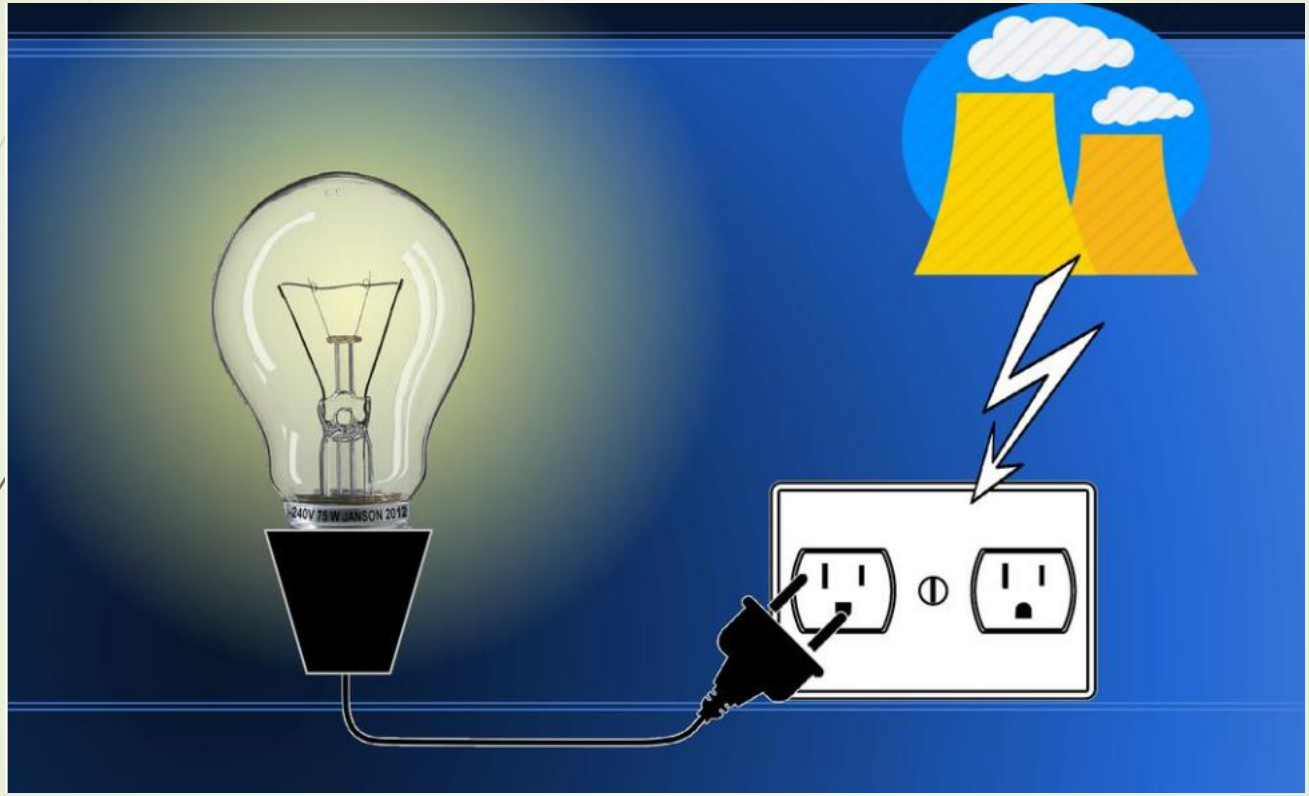
Définition du mot API

En informatique, **API** est l'acronyme d'**Application Programming Interface**, que l'on traduit en français par **interface de programmation applicative** ou **interface de programmation d'application**.

L'API peut être résumée à une solution informatique qui permet à des applications de communiquer entre elles et de s'échanger mutuellement des services ou des données.

Il s'agit en réalité d'un ensemble de fonctions qui facilitent, via un langage de programmation, l'accès aux services d'une application.

Définition du mot API



Les API JavaScript de HTML5

- Le JavaScript côté client a de nombreuses API à sa disposition.
- Elles ne font pas partie du langage JavaScript lui-même,
- elles sont construites par-dessus JavaScript, offrant des super-pouvoirs supplémentaires à utiliser dans un code.
- Elles se répartissent en deux catégories :
  - **Les API web**
  - **Les API tierces**

Les API JavaScript de HTML5

- **Les API web** sont intégrées au navigateur web et permettent de rendre disponibles les données du navigateur et de son environnement afin de réaliser des choses complexes avec. Ainsi, l'API *Web Audio* fournit des constructions JavaScript pour manipuler des données audio dans le navigateur.

  On pourra utiliser cette API afin de récupérer une piste audio puis d'abaisser son volume, d'y appliquer des effets, etc. Sous le capot, c'est le navigateur qui s'occupe des couches plus complexes en code « bas niveau » (C++ ou Rust par exemple) afin de réaliser le traitement du signal. Là encore, cette complexité est masquée par l'abstraction offerte par l'API.

Les API JavaScript de HTML5

- **Les API tierces** ne sont pas intégrées au navigateur par défaut, et vous devez généralement récupérer le code de l'API et des informations depuis un site web. Par exemple : l'API Twitter vous permet d'afficher vos derniers tweets sur votre site web. Elle fournit un ensemble de constructions que vous pouvez utiliser pour interroger le service Twitter et qui renvoie alors les informations demandées.

Les API JavaScript de HTML5

- Le **HTML5** propose une série de nouvelles API, pour la plupart sous JavaScript, qui peuvent être implémentées sur tous les navigateurs. Voici quelques exemples d'API ajoutées pour HTML5 :
- dessin en 2D via la balise <canvas> et le SVG et ajout de contenu 3D sur les pages avec les API tierces WebGL et Khronos Group
- géolocalisation
- activation des contenus audio et vidéo via les balises <audio> et <video>, également nouvelles
- applications hors connexion : l'Application cache
- exécution de tâches parallèles au code de la page. Cette API peut interagir avec le script principal de la page.
- l'édition de contenus, qui fonctionne avec le nouvel attribut <contenteditable>
- glisser-déposer activée sous l'attribut <draggable>
- un web storage étendu, qui ne remplace pas les cookies mais offre une amplitude largement supérieure aux versions précédentes
- Etc,….

Le multimédia (vidéo, audio)

<video>

- **Principe de base :**

```
<video width="320" height="240" controls="controls">
  <source src="movie.mp4" type="video/mp4" />
  <source src="movie.ogg" type="video/ogg" />
  Votre navigateur ne supporte pas le tag video.
</video>
```

- L'attribut « controls » ajoute les boutons de contrôle (play, stop, etc.)
- Le premier format supporté sera lu !

- Démonstration :
  https://www.w3schools.com/tags/tryit.asp?filename=tryhtml5_video

Les formats video

NAVIGATEUR	H.264/MP4	OGG THEORA	WEBM
Mozilla Firefox	non	3.5	4.0
Opera	non	10.5	10.6
Microsoft Internet Explore	9.0	non	oui si codecs installés
Google Chrome	4.0 à 16.0	4.0	6.0
Apple Safari	5.0	non	non

Le tag <video> : les attributs

- **controls** donne accès aux contrôles de lecture (boutons de navigation, volume, etc, selon les possibilités du navigateur), ou les masque s'il est omis.

- **preload="auto"** permet de de spécifier au navigateur de **débuter le téléchargement** de la vidéo tout de suite, en anticipant sur le fait que l'utilisateur lira la vidéo. Attention, cette option est à manier avec prudence (il est préférable que ce soit la seule raison d'être de la page).

- **autoplay="true"** comme son nom l'indique, permet de lancer la **lecture automatiquement**

- **poster="image.jpg"** permet d'indiquer une **image à afficher par défaut** dans l'espace réservé par la vidéo, avant que la lecture de celle-ci ne soit lancée

- **loop** indique que la lecture doit s'effectuer **en boucle**.

<video> : contrôle depuis le DOM

- <video> possède des **méthodes**, des **propriétés** et des **événements** que l'on peut manipuler depuis JavaScript et l'API du DOM.
- **Méthodes** pour jouer, mettre en pause etc.
- **Propriétés** (durée, position courante, etc.) que l'on peut consulter/modifier
- Des **événements** sont générés durant le cycle de vie et on peut définir des écouteurs. On peut aussi envoyer des événements pour contrôler le lecteur vidéo.

<video> est un élément du DOM

- On peut le manipuler ou le créer depuis du code, via l'API de manipulation du DOM

```
var video = document.createElement('video');
video.src = 'video.ogv';
video.controls = true;
document.body.appendChild(video);
```

<video> exemple


```
<video style='opacity: 0.5;
      position: absolute;
      top: 55%;
      left: 40%;
      width: 320px; height:240px;
      z-index: 1'
      src=‘mavideo.ogg'
      width='360' height='240'
      id="vid">
  <p>This video cannot be displayed. <br/>
   No support for HTML5?</p>
</video>

<span id='play' onclick="playVideo();">play</span>
<span id='pause' onclick="pauseVideo;">pause</span>
<span id='rewind' onclick="rewindVideo();">Retour à la position de départ</span

<script>
    vid = document.getElementById("vid");
    play = document.getElementById("play");
    pause = document.getElementById("pause");
    rewind = document.getElementById("rewind");
</script>
```

<video> exemple


```
function playVideo() {
  vid.play();
  play.style.color='black';
  pause.style.color='#999';
  rewind.style.color='#999';
}

function pauseVideo() {
  vid.pause();
  play.style.color='#999';
  pause.style.color='black';
  rewind.style.color='#999';
}

function rewindVideo() {
  vid.currentTime = 0;
  play.style.color='#999';
  pause.style.color='#999';
  rewind.style.color='black';
}
```

<video> attributs du tag

- **src** : source de la vidéo
- **width et height** : dimension de la vidéo. SI non spécifiés, valent la largeur et la hauteur du fichier vidéo. Si vous spécifiez une dimension mais pas l'autre, le navigateur va ajuster la taille de la dimension non spécifiée afin de préserver les proportions de la vidéo.
- **controls** : Si cet attribut booléen est présent, le navigateur affiche ses propres contrôles vidéo pour la lecture et le volume. Si non présent la première image est affichée (ou l'image poster spécifiée). Lecture via événement JavaScript seulement.
- **poster** L'attribut poster permet de spécifier une image que le navigateur utilisera alors que la vidéo est en cours de téléchargement, ou jusqu'à ce que l'utilisateur commence la lecture de la vidéo. Si cet attribut n'est pas spécifié, la première image de la vidéo sera utilisée à la place

<video> attributs du tag

- **autoplay** Spécifie au navigateur de lancer la lecture de la vidéo automatiquement.
- **autobuffer** : spécifie au navigateur de commencer le téléchargement de la vidéo tout de suite, en anticipant sur le fait que l'utilisateur lira la vidéo. (Cette partie de la spécification est actuellement en pleine mutation et sujette à changement)
- **loop** : un autre attribut booléen, indique de lire la vidéo en boucle.
- Tous les attributs ne sont pas encore supportés par tous les navigateurs, mais les plus courants le sont.

<video> table des propriétés, méthodes, événements

- Certaines propriétés ne sont disponibles qu'après le chargement du début de la vidéo

Methods	Properties	Events
play()	currentSrc	play
pause()	currentTime	pause
load()	videoWidth	progress
canPlayType	videoHeight	error
	duration	timeupdate
	ended	ended
	error	abort
	paused	empty
	muted	emptied
	seeking	waiting
	volume	loadedmetadata
	height	
	width	

<video> table des propriétés, méthodes, événements

La méthode **canPlayType()** permet de tester si le navigateur peut lire un type donné de vidéo
Les valeurs retournées sont :

- "probably " : le navigateur prend probablement en charge ce type audio/video
- "maybe" : le navigateur ne prend pas en charge ce type audio / vidéo

<video> autre exemple interactif

- Modification de la taille de la vidéo, pendant qu'elle joue, depuis JavaScript

<video> et CSS

- Tous les styles CSS, notamment les effets de transition, transformations géométriques et animation de CSS3 s'appliquent
- video {
  width: 75px;
  -o-transition: all 0.5s ease-in-out;
  -webkit-transition: all 0.5s ease-in-out;
  -moz-transition: all 0.5s ease-in-out;
  transition: all 0.5s ease-in-out;
  }
  video:hover, video:focus { width:600px; }

Le tag <audio>

- Similaire au tag <video>

```
<audio controls="controls">
  <source src="">
```

- Comme pour <video> l’attribut controls ajoute des boutons de contrôle

Le tag <audio>

```
<audio controls="controls">
  <source src="song.ogg" type="audio/ogg" />
  <source src="song.mp3" type="audio/mpeg" />
  Your browser does not support the audio element.
</audio>
```

Les formats audio

Pour diffuser de la musique ou n'importe quel son, il existe de nombreux formats. La plupart d'entre eux sont compressés (comme le sont les images JPEG, PNG et GIF) ce qui permet de réduire leur poids :

•**MP3** : vous ne pouvez *pas* ne pas en avoir entendu parler ! C'est l'un des plus vieux, mais aussi l'un des plus compatibles (tous les appareils savent lire des MP3), ce qui fait qu'il est toujours très utilisé aujourd'hui.

•**AAC** : utilisé majoritairement par Apple sur iTunes, c'est un format de bonne qualité. Les iPod, iPhone et autres iPad savent les lire sans problème.

•**OGG** : le format Ogg Vorbis est très répandu dans le monde du logiciel libre, notamment sous Linux. Ce format a l'avantage d'être libre, c'est-à-dire qu'il n'est protégé par aucun brevet.

•**WAV** (**format non compressé**) : évitez autant que possible de l'utiliser car le fichier est très volumineux avec ce format. C'est un peu l'équivalent du Bitmap (BMP) pour l'audio.

La compatibilité dépend des navigateurs, mais elle évolue dans le bon sens au fil du temps. Pensez à consulter Caniuse.com pour connaître la compatibilité actuelle du MP3, AAC, OGG, WAV...

Le tag <audio> : attributs

Attribut	Valeur	Description
autoplay	*autoplay*	Indiquez si la lecture doit se lancer automatiquement
controls	*controls*	Indiquez si les boutons de controls (lecture, arret ...) doivent apparaitre
loop	*loop*	Indiquez si le fichier doit redémarrer (boucle) lorsque sa lecture est terminé
preload	*auto* *metadata* *none*	Indiquez si et comment les fichiers seront pré-chargés
src	*url*	Indiquez l'adresse de voter fichier audio

<audio> et le DOM

```
var audio = document.createElement('audio');
audio.src = 'audio.oga';
audio.controls = true;
document.body.appendChild(audio);
```

- Ou encore, on peut utiliser un constructeur :

```
var audio = new Audio('audio.oga');
```

- Ou bien

```
var audio = new Audio();
audio.src = 'audio.oga';
```

Codecs supportés

- Le mp3 n'est pas supporté partout par exemple

Browser	MP3	Wav	Ogg
Internet Explorer 9	YES	NO	NO
Firefox 4.0	NO	YES	YES
Google Chrome 6	YES	YES	YES
Apple Safari 5	YES	YES	NO
Opera 10.6	NO	YES	YES

Attributs, méthodes et événements

- Attributs : les mêmes que pour le tag vidéo à part « poster » : src, preload, autoplay, loop, controls
- Les méthodes sont les mêmes : play(), pause(), load()…
- Les événements sont les mêmes que ceux du tag vidéo, à 99%. On parle en fait de « media events » HTML5
  - Ex : onload, onpause, oncanplay, etc.

WebGL

WebGL est un standard pour la programmation en 3D avec le navigateur comme plateforme. La spécification finale du standard a vu le jour en 2010 et est définie par le Khronos Group. Il permet de réaliser des animations, des interfaces ou des jeux en 3D fonctionnant à la fois en ligne et hors connection.

- https://www.scriptol.fr/programmation/webgl.php