RFC 9239 : Updates to ECMAScript Media Types

Stéphane Bortzmeyer
<stephane+blog@bortzmeyer.org>

Première rédaction de cet article le 11 mai 2022

Date de publication du RFC : Mai 2022

https://www.bortzmeyer.org/9239.html

Ce RFC documente les types de médias pour le langage de programmation JavaScript (dont le nom officiel est ECMAScript, qu'on retrouve dans le titre de ce RFC). Il remplace le RFC 4329[1] et le principal changement est que le type recommandé est désormais `text/javascript` et non plus `application/javascript`.

Si vous voulez vous renseigner en détail sur JavaScript, notre RFC recommande de lire la norme ECMA, en . Cette norme est développée en dehors de l'IETF et le choix des types de médias (aussi appelés types MIME, cf. RFC 2046) n'est donc pas forcément en accord avec les règles de l'IETF (RFC 6838). C'est pour cela que l'IESG a ajouté une note d'avertissement au RFC. Mais, bon, ce n'est pas trop grave en pratique. Le type recommandé est donc désormais `text/javascript`. D'autres types existent, `application/ecmascript`, `application/javascript`, etc, mais ils sont maintenant considérés comme dépassés.

Il existe plusieurs versions de la norme JavaScript et d'autres apparaitront peut-être dans le futur. Mais le type officiel n'indique pas de version (il a existé des propositions comme `text/javascript1.4`) et compte que toutes ces versions sont suffisamment compatibles pour qu'on ne gère pas la complexité d'avoir plusieurs types. Normalement, ECMA s'engage à ne pas bouleverser le langage.

Le choix de `text/` est contestable car la définition originale de `text/` dans le RFC 2045 prévoyait plutôt `application/` pour les programmes. Le RFC 4329 enregistrait les types `text/javascript` et `application/javascript`, et recommandait le `application/javascript` (en accord avec ce que dit le RFC 6838). Aujourd'hui, c'est le contraire, `text/javascript` est le préféré. D'une manière

1. Pour voir le RFC de numéro NNN, `https://www.ietf.org/rfc/rfcNNN.txt`, par exemple `https://www.ietf.org/rfc/rfc4329.txt`

générale, `application/` n'a guère été utilisé (pas seulement dans le cas de JavaScript). (Personnellement, cela me convient très bien : pour moi, si ça peut s'afficher avec cat et s'éditer avec vi, c'est du texte.)

JavaScript est donc officiellement du texte, et la section 4 de notre RFC précise : du texte en UTF-8. Le paramètre `charset` est donc facultatif (même si le RFC 6838 dit le contraire). Si vous aimez les détails d'encodage, la section 4 vous ravira (c'est l'un des points qui a suscité le plus de discussion à l'IETF).

Le type `text/javascript` est enregistré à l'IANA <https://www.iana.org/assignments/media-types/text/javascript> ; les types comme `application/javascript` sont marqués comme dépassés <https://www.iana.org/assignments/media-types/application/javascript>. Même chose pour `text/ecmascript`. Le nom officiel de JavaScript est ECMAscript, puisque normalisé à l'ECMA mais personne n'utilise ce terme. (Il faut quand même noter que JavaScript est un terme publicitaire mensonger puisqu'il avait été choisi par le marketing pour profiter de la popularité - à l'époque - de Java, un langage avec lequel il n'a rien à voir.) Enfin, les types comme `text/x-javascript` qu'on voit parfois trainer datent d'avant le RFC 6648, qui abandonnait les identificateurs semi-privés commençant par `x-`.

La section 5 couvre les questions de sécurité. Elle est très longue car l'envoi de JavaScript à un programme qui l'exécutera (ce qui est très courant sur le Web) pose plein de problèmes de sécurité. Le RFC rappelle que l'exécution automatique d'un programme fourni par un tiers est évidemment intrinsèquement dangereuse, et **doit** se faire dans un bac à sable. Parmi les dangers (mais il y en a beaucoup d'autres !) le déni de service puisque JavaScript est un langage de Turing et permet donc, par exemple, des boucles infinies.

L'annexe B du RFC résume les changements depuis le RFC 4329 : le principal est bien sûr l'abandon de `application/javascript` au profit de `text/javascript`. Il y a aussi l'ajout de quelques détails comme une faille de sécurité nouvelle, et le cas des modules JavaScript.

Ah, un point de détail cité au détour d'un paragraphe du RFC : il n'y a pas de norme pour les identificateurs de fragments dans un URI pointant vers une ressource de type `text/javascript`. Si `https://code.example/js/foreach.js#l23` pointe vers du JavaScript, la signification du `#l23` (l'identificateur de fragment) n'est pas spécifiée.