Inclure jQuery de manière optimisée

ASTUCE javascript

Publié par Rodolphe le 21 Juin 2010, mis à jour le 24 Janvier 2017 (376848 lectures)

javascript jquery framework api script

jQuery est un framework JavaScript très célèbre, permettant en quelques lignes de code de dynamiser un site web, de créer de petites animations, des interactions avec les formulaires, de programmer des appels Ajax. Voici, en détails, quelques possibilités d'intégration à votre site, afin de minimiser l'impact sur les performances et le temps de chargement.

Appel classique à la librairie (fichier téléchargé)

De la manière la plus basique, il est vous est proposé sur le site jQuery.com de télécharger le fichier - en affichant son code source - et de le placer sur votre site, par exemple dans `jquery.js` ou `jquery-x.x.x.min.js`.

Fichier et compression

Vous récupérerez ainsi une version "minifiée", c'est à dire compressée jusqu'à un certain niveau par un algorithme supprimant espace et commentaires, mais rendant peu lisible la déclaration des fonctions. Ceci n'est pas bien grave car il s'agit la plupart du temps de ne pas modifier cette librairie mais de la combiner à nos propres fonctions, déclarées de façon externe.

```
/*!
 * jQuery JavaScript Library v1.4.2
 * http://jquery.com/
 *
 * Copyright 2010, John Resig
 * Dual licensed under the MIT or GPL Version 2 licenses.
 * http://jquery.org/license
 *
 * Includes Sizzle.js
 * http://sizzlejs.com/
 * Copyright 2010, The Dojo Foundation
 * Released under the MIT, BSD, and GPL Licenses.
 *
 * Date: Sat Feb 13 22:33:48 2010 -0500
 */
(function(A,w){function ma(){if(!c.isReady){try{s.documentElement
e(a[0],b):w}function J(){return(new Date).getTime()}function Y(){
j.length;n<r;n++)for(k=0;k<u.length;k++){i=u[k];if(j[n].selector=
"&")}function qa(a){return!a||!a.parentNode||a.parentNode.nodeTyp
true;if(j=c.fragments[a[0]])if(j!==1)f=j}if(!f){f=b.createDocumen
Wa=/^(\s|\uOOAO)+|(\s|\uOOAO)+$/g,Xa=/^<(\w+)\s*\/?>(?:<\/\1>)?$/
(d[1]||!b))if(d[1]){f=b?b.ownerDocument||b:s;if(a=Xa.exec(a))if(c
a)}else return!b||b.jquery?(b||T).find(a):c(b).find(a);else if(c.
```

Voyez les différences entre la version non minifiée et la version minifiée. Sachez qu'il est possible de faire de même pour vos propres scripts grâce à Google Closure Compiler ou encore YUI Compressor de Yahoo.

💡 Il vous est aussi possible de délivrer ce fichier re-compressé au format gzip pour gagner encore en taille, c'est à dire quelque petites dizaines de Ko selon la version, au lieu de de la centaine prévue. Pour ceci, consultez le tutoriel correspondant : Compression des pages web avec Gzip ou Deflate en HTTP.

Performances au chargement

Pour des questions de performance, Google Code et Yahoo Developer Network : Exceptional Performance recommandent :

- de placer les appels aux librairies en fin de page avant `</body>`
- de placer les appels aux feuilles de style CSS avant, dans la section `<head>` grâce au tag `link`

- de combiner au maximum les fichiers JavaScript/CSS, de les compresser/minifier et de les externaliser (c'est à dire les placer chacun dans un fichier externe à la page et non en ligne avec toutes leurs instructions dans le code html)

Vous obtiendrez ainsi :

```HTML
<!doctype html>
<html>

  <head>
     <title>Titre de la page</title>
     <link rel="stylesheet" type="text/css" href="styles.css">
  </head>

  <body>
     ...
     <script src="jquery.js"></script>
     <script src="mon_script.js"></script>
  </body>

</html>
```

Depuis un CDN de type Google

Google propose un CDN (Content Delivery Network) dédié au chargement de différentes librairies. L'avantage étant de bénéficier d'un hébergement externe et rapide (mais soumis la plupart du temps à une requête HTTP+DNS supplémentaire) avec un fichier délivré en gzip sans avoir à configurer votre serveur.

```JavaScript
<script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js">
</script>
```

D'autres librairies sont disponibles. Reprenons notre exemple, sans avoir besoin d'héberger le fichier `jquery.js` :

```HTML
<!doctype html>
<html>
```

```
  <head>
     <title>Titre de la page</title>
     <link rel="stylesheet" type="text/css" href="styles.css">
  </head>

  <body>
     ...
     <script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.1.0/jquery.min.js">
</script>
     <script src="mon_script.js"></script>
  </body>

</html>
```

Votre premier script jQuery

Bien entendu il vous faudra placer votre propre fichiers JavaScript ou vos déclarations à la suite de cet appel, dans `mon_script.js`. Ceci est la plupart du temps valable pour les autres librairies JavaScript.

Pour ceci vous avez deux possibilités :

- placer une fonction directement dans le code de la page à l'aide de la balise `<script>` mais sans fichier externe (sans attribut `src`). Le seul avantage de cette méthode est de supprimer une requête HTTP, mais cela ajoute du contenu à votre code HTML qui ne sera pas (ou plutôt rarement) mis en cache.
- **placer vos instructions dans un fichier externe** (*.js), plus facile à maintenir et à inclure sur un groupe de pages de votre site, avec le petit inconvénient d'effectuer une requête HTTP en plus.

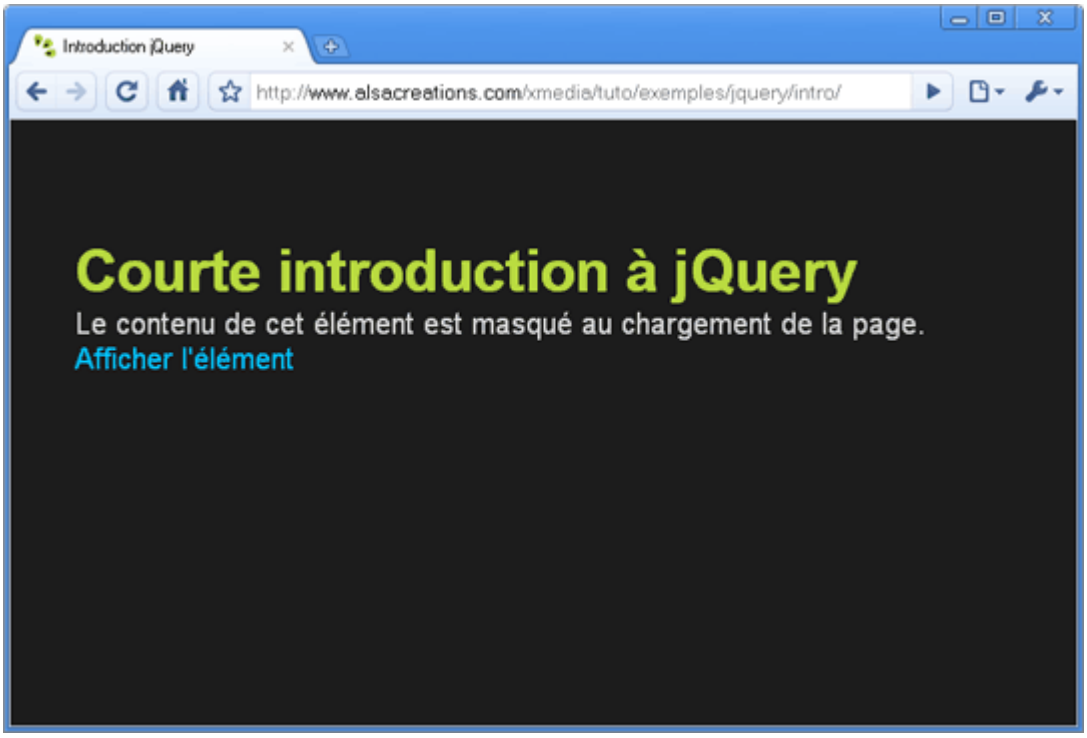
Utilisons la deuxième solution pour placer notre script jQuery dans `mon_script.js`. Dans la majorité des cas, nous souhaiterons exécuter le code au chargement de la page, grâce à la fonction `ready()` appliquée à l'objet `document`. C'est-à-dire lorsque les objets de la page seront prêts à être manipulés par JavaScript.

```JavaScript
jQuery(document).ready(function($) {
   // Votre code ici avec les appels à la fonction $()
});
```

La **fonction `$()` est la fonction magique de jQuery**. Celle-ci permet de cibler les éléments grâce aux sélecteurs, de déclarer des gestionnaires d'événements, d'utiliser les autres fonctions de la librairie, il s'agit en réalité d'un alias sur l'objet jQuery lui-même.

Tranchons dans le gras

Le but de cet astuce étant juste de faire appel minimaliste à jQuery, nous nous contenterons pour cette démonstration de masquer un élément `<div id="masque">`, puis de le révéler au clic sur un lien `<a id="afficher">Afficher l'élément</a>`.



Toujours dans la déclaration `$(document).ready(function() {`, nous placerons les instructions jQuery de notre script :

```javascript
$(document).ready(function() {

   // Dès le chargement on masque l'élément portant l'id #masque
   // grâce à la fonction hide() de jQuery
   $("#masque").hide();

   // On déclare un gestionnaire d'événement 'click' sur un lien
   // pour afficher l'élément précédemment masqué
   $("a#afficher").on("click",function() {
```

```
        // La fonction click() appliquée à notre sélecteur $("a#afficher")
        // prend en argument une fonction anonyme (sans nom) contenant
        // le reste des instructions :

        $("#masque").show("fast");

        // L'argument "fast" est facultatif mais nous permet
        // d'afficher l'élément avec une petite animation

        return false;

        // On retourne 'false' pour prévenir l'exécution du lien
        // c'est à dire pour éviter au navigateur de changer de
        // page en suivant son attribut href

    });

    // Faites bien attention à la syntaxe et à l'imbrication des
    // parenthèses et accolades

});
```

Démo

Consultez la démonstration pour explorer le code source :

- **index.html**
- **mon_script.js**

Voici donc une brève introduction à l'inclusion de jQuery sur votre site. N'hésitez pas à consulter la documentation de jQuery pour faire connaissance avec les autres fonctions et sélecteurs possibles ou à utiliser la section JavaScript du forum.

Quiz

Testez vos connaissances avec un Quiz JavaScript

Commentaires

Ladytron

21 Juin 2010 à 20h35

Déplacer les javascript en fin de body et non dans le head peut être intéressant. Pourquoi pas, c'est à essayer.
Si cela permet effectivement d'accélérer significativement le chargement des pages, c'est une bonne alternative.

jpvincent

22 Juin 2010 à 08h30

oui, c'est ce que j'ai fini par faire sur ma HomePage, le gain apparent en réactivité a été hallucinant. Je dis apparent car la page totale s'affichait à peine plus vite, par contre l'utilisateur voyait le HTML quasi immédiatement alors qu'avant il devait attendre que le JS soit là.
Après le problème pour beaucoup de développeurs est qu'ils ont du JS inline qui "casse" si le javascript n'est pas dans le <head>. Pour ça j'ai trouvé une solution :
http://jpv.typepad.com/blog/2010/05/performan...
que j'ai appelé mute + eval

pfoofen

22 Juin 2010 à 10h28

On peut aussi remplacer :
$(document).ready(function(){
//notre script
}
Par :
$(function(){
//notre script
}

Oli78
22 Juin 2010 à 10h33

Question bête, pourquoi utiliser "return false;" et pas "e.preventDefault()" ?

Tomas
22 Juin 2010 à 10h37

Sinon,pour plus d'info sur les plus et les moins (surtout en termes de perfs) d'externaliser ses librairies JS chez Yahoo! ou Google, le blog performance vaut le détour
http://performance.survol.fr/2009/02/cache-ce...

grunky
22 Juin 2010 à 10h39

@dew "mais soumis la plupart du temps à une requête HTTP+DNS supplémentaire"

On fait un hit de plus mais la requête DNS est plutôt un avantage puisqu'elle va permettre de paralléliser les téléchargement de fichier.
Si par exemple on à jquery qui se charge depuis google et function.js qui se charge depuis notre domaine , les deux se feront en même temps alors que si les deux avait été sur notre domaine il aurait fallu attendre la fin du chargement de jquery pour charger function.js
Google et Yahoo préconise d'ailleurs ce genre de pratique (sans toutefois en abuser) pour accélérer le chargement des pages.

Skoua
22 Juin 2010 à 10h59

J'ai toujours cru que ce n'était pas correct de mettre du script en-dehors de body.

On peut donc inclure ses fichiers js avant la fin de body sans problème ?

Nickko
22 Juin 2010 à 12h03

@Skoua
en dehors du body, si c'est dans le head, ça va mais après ça fermeture, effectivement ce n'est pas

correcte.

Nickoeuh

22 Juin 2010 à 12h44

@pfoofen : Quel est l'avantage de $(document).ready(function(){ part rapport à $(function(){ ?

Florent V.

22 Juin 2010 à 13h36

@Nickoeuh, l'avantage principal est que c'est plus explicite. :)
Sinon, aucune différence. On utilisera l'un ou l'autre selon qu'on préfère quelque chose de plus long mais explicite, ou un raccourci plus cryptique mais, euh, plus court.

Skoua

22 Juin 2010 à 14h12

@Nickko : Heu... J'ai pas compris. :p

dew

22 Juin 2010 à 15h01

@Oli78 : Parce que détailler le pourquoi du comment aurait alourdi l'article, dont ce n'était pas l'objet. Mais oui, on peut tout à fait utiliser preventDefault (si l'événement a été reçu en argument).

Quant à $(function(){ ... }) il s'agit juste d'un raccourci d'écriture.

MatTheCat

22 Juin 2010 à 15h02

Je n'ai jamais compris pourquoi mais sur un site le fait de charger JQuery avant provoquait l'apparition d'un scroll horizontal..!
Et j'ai lu quelque part que le téléchargement d'un script se faisait "seul" (sans parallélisation) , est-ce vrai ?
De plus il me semblait que le DOM était construit lorsque </body> est atteint ; doit-on tout de

même utiliser la fonction magique de JQuery ?

Merci de m'éclairer ^^

Nickko

22 Juin 2010 à 15h18

@Skoua :
1. En dehors du body, c'est correcte que si c'est dans le head, ça tu comprends ?
2. En dehors du body, ce n'est donc pas correcte si ce n'est pas dans le head. Tu comprends ?

Que n'as-tu pas compris ?

wrb3os

22 Juin 2010 à 15h46

Il existe aussi pour insérer un script, scriptsrc.net, il permet facilement de pouvoir utiliser une des nombreuses librairies javascript facilement, sans avoir pour autant à télécharger la dite librairie souhaitée.

Skoua

22 Juin 2010 à 16h39

@Nickko : En gros c'est soit le head soit le body ?
Ok j'ai compris. :D

dj DMSR

22 Juin 2010 à 16h52

Dîtes-moi si je me trompe:
-dans le head il n'y a pas de parallélisation des chargements (.js comme .css), ils sont séquencés dans l'ordre d'apparition dans le code. Et le rendu de la page ne se fait qu'après chargement de tous les .js et .css.
-dans le body il y a parallélisation. Le rendu de la page se fait avant le chargement des .js -puisqu'ils sont les derniers chargés-, et donc on a l'impression d'un chargement plus rapide. Mais on risque donc d'avoir des chargements qui se terminent dans un ordre différent de celui des

appels dans le code. En particulier avec les frameworks js comme jQuery qui sont en général plus lourds que les ou l'autre(s) script(s) [On a quand même souvent intérêt de séparer jQuery du reste pour le mettre en cache afin d'accélérer encore le chargement]. OR il y a dépendance entre le framework et les ou l'autre script(s), puisque -par exemple- le gestionnaire d'événement de la page dépend du framework. On peut donc avoir un gestionnaire chargé avant le framework. Et donc bug.
OR il me semble avoir lu que seule la DERNIÈRE version de jQuery résolvait ce problème (et pas avec toutes les -vieilles- versions de navigateurs). Personnelllement je sais que mettre prototype.js avant la fermeture du body ne fonctionne pas!

Note: charger les scripts dans le body revient à faire un chargement asynchrone

références: http://happyworm.com/blog/tag/labjs/
"The good news is that jQuery1.4 now checks for the dom-ready state and doesn't just rely on the dom-ready event to detect when the DOM is ready, meaning that we can now use a simpler solution with more confidence."
-> http://github.com/jquery/jquery/blob/master/s...

dj DMSR
22 Juin 2010 à 16h58

À lire absolument le blog du guru de la performance web: Steve Souders
http://www.stevesouders.com/blog/2009/04/27/l...

Capripot
22 Juin 2010 à 17h17

J'ai du lire y'a pas mal de temps, que c'était "mieux et plus standard" de mettre les librairies dans le head, ce n'est plus valable ?

jpvincent
23 Juin 2010 à 08h16

@Capripot : si, le <head> est fait pour ça en fait. Mais cela reste valide de mettre ça dans le <body>, et tes utilisateurs préfèrent aller sur un site qui donne l'impression de vitesse plutôt qu'un site valide

jpvincent

23 Juin 2010 à 08h34

@dj DMSR : le comportement en plaçant css/js dans le head ou le body est exactement le même : ça bloque le rendu (page blanche ou page affichée jusqu'à hauteur de ton include) et le téléchargement des ressources trouvées après ton include. C'est pour ça qu'il est conseillé de mettre ton include JS en fin de body, le browser pendant ce temps là télécharge aussi les images inline et de CSS

Pour ce qui est de l'ordre des dépendances, le browser est intelligent : si la ressource qui vient en 2nd dans la source arrive plus vite, elle n'est pas exécutée avant que la première ne soit arrivé.

Super_baloo8

23 Juin 2010 à 19h01

Génial cette petite astuce permet d'accélérer l'affichage de la page de manière vraiment impressionnante !

Milles merci !

Chrisis

23 Juin 2010 à 19h55

Concernant l'appel des JS avant la fin body pour "accélérer" l'affichage de la page, on est d'accord, la page s'affiche juste plus rapidement car les JS sont appelés plus tard.

Mais si dans mes scripts JS, j'injecte du contenu dans ma page au domready (call AJAX par exemple ou autres), lorsque la page va s'afficher, je n'aurais donc pas encore mon contenu injecté dans la page il me semble.

ça ne m'a pas l'air utilisable n'importe quand cette solution après il doit y avoir moyen de bidouiller tout ça mais je suis pas tout à fait convaincu !

dew

24 Juin 2010 à 12h12

C'est bien pour cela que je précise dans le contenu qu'il s'agit de placer les autres scripts après la librairie, et bien entendu de gérer la situation en fonction de ce que réalisent ces scripts. Dans l'absolu si le site est bien conçu il ne devrait pas y avoir de document.write à l'arrache un peu partout comme on le voyait il y a 10 ans. Autre cas de figure : le fameux script html5.js pour IE (qui effectue des document.createElement pour les nouveaux tags) doit bien entendu être placé en début de page avant l'apparition de ces éléments.

Florent V.

24 Juin 2010 à 12h15

@Chrisis, si tu crées des éléments dans le DOM au domready, tu as forcément un risque que ta page s'affiche sans ces éléments. Si en plus tu récupères le contenu à injecter en Ajax, tu as droit au temps de ta requête Ajax (qui comprend latence réseau, temps de réponse du serveur, temps de chargement, parsing de ta réponse en JSON ou autre…), donc ce risque augmente très fortement. Si enfin tu appelles ta librairie JS à la fin du document plutôt que dans le HEAD, la principale différence va être que l'affichage de la page ne va pas être bloqué (page blanche) pendant le chargement du JS, donc le temps que tu gagnes sur l'affichage se rajoute au laps de temps où ton DOM n'est pas encore modifié.

Si tu veux éviter ce genre de problème, il faut avoir le contenu directement dans le code HTML. Ou bien se faire une raison et afficher la page avec un indicateur de chargement par exemple.

dj DMSR

24 Juin 2010 à 22h34

@jpvincent : pas sûr! mon anglais n'est pas terrible mais si j'ai bien compris ce qu'il est dit sur les pages dont les liens se trouvent dans mes commentaires, il y a bien une différence entre head/body. Et le navigateur n'est si intelligent... -> gros doute?!
Voir par ex ce super outil du Guru Steve Souders pour faire ses propres tests:
http://stevesouders.com/cuzillion/
et surtout lire ses explications

dj DMSR

24 Juin 2010 à 22h42

@Florent V. : le problème avec jQuery avant la fermeture du body c'est: DOM est-il bien "ready" (enfin je veux dire la fonction jQuery est-elle prête à fonctionner correctement) quand j'ai chargé mon gestionnaire d'événements dans un fichier à part... Il semble que oui depuis la dernière

version
Voir par ex: http://www.nczonline.net/blog/2009/07/28/the-...
ou les efforts de @getify de pour gérer les dépendances: http://labjs.com/

emma1979
28 Juin 2010 à 17h30

waouww, ca marche en plus! Merci!

dldstyle
29 Juin 2010 à 09h09

Je ne connaissais pas cette possibilité offerte par Google et ses API. J'imagine en plus que outre le téléchargement en parallèle de la page, si beaucoup de sites utilisent cette possibilité, le fichier jquery devrait être dans le cache de beaucoup d'internautes et rendre le chargement de la page encore plus rapide, non ?

jpvincent
29 Juin 2010 à 10h08

@dj DMSR : avant de te répondre, j'avais justement fait un tour sur cuzillion, pour confirmer ce que je pensais :) Sinon je n'ai jamais entendu parler d'une différence entre head et body, mais si tu arrives à retrouver quelque chose à ce sujet, je prends toute info !
et la raison pour laquelle le browser bloque tout rendu dès qu'on met une balise SCRIPT, c'est justement parce qu'il suppose que le JS peut affecter la page, donc il ne veut pas faire le boulot en double. Et à côté de ça il exécute les JS dans l'ordre de la source, quel que soit l'ordre d'arrivée du téléchargement.
Ca fait un bail que je n'ai plus de JS qui se suivent, donc peut être que ma mémoire me joue des tours, mais c'est pratiquement sur

Tony Monast
29 Juin 2010 à 17h39

@dldstyle : En effet, le fichier est mis en cache et est réutilisé par les sites appelant la même librairie.

Toutefois, je suis personnellement mal à l'aise d'utiliser une librairie Javascript hébergée ailleurs. Par mesure de sécurité ou excès de prudence, je préfère de loin les héberger moi-même, car même si certains ont une confiance inébranlable face au géant Google, qui sait si un jour un comique ne modifierait pas cette librairie pour y injecter du code malicieux. Dans un tel cas, ça ferait un beau bordel sur tous les sites faisant appel à cette librairie.

D'autant plus que l'application ou le site dépend maintenant d'un tiers service pour fonctionner tel que prévu.

stb007

22 Juillet 2010 à 14h36

Je pense que la manière la plus optimisée d'inclure du JS est de le coller en fin de page mais inline, pas en inclusion, déjà on économise une requête HTTP, de plus coté serveur & client ça ne fait qu'un fichier à compresser/décompresser (deflate/gzip) maintenant ça doit dépendre de votre manière de concevoir un site, moi j'utilise le zend Framework et je met en place un maximum de JS dans les vues, je trouve ça pratique, plutôt que d'avoir des tonnes de fichiers dans un répertoire JS, au finale c'est le bordel et on s'y retrouve pas, alors que le JS spécifique à une vue placé dans la vue elle même est bien plus pratique.
Par contre ok avec l'article sur l'implémentation des librairies communes (Jquery & cie)

@+
Laurent

Tony Monast

23 Juillet 2010 à 18h03

stb007, en mettant le code inline, on perd alors l'avantage de la mise en cache du fichier .js par le navigateur Web. C'est loin d'être optimisé.

Alsacréations est une communauté dédiée à la conception de sites et applications web de qualité, grâce aux standards W3C, aux feuilles de styles CSS, aux langages HTML et JavaScript, et à l'accessibilité. Réalisé par l'agence web Alsacreations.fr · Hébergement Syazen · À propos et mentions légales · Données personnelles

https://www.alsacreations.com/