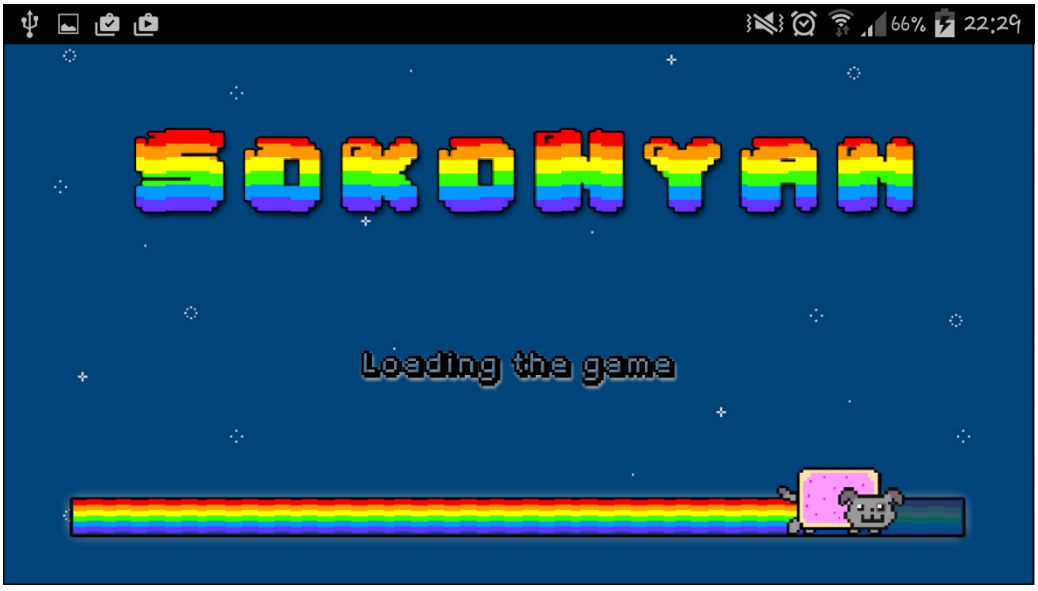
OPTIMISER LES PERFORMANCES



07/06/2017

Projet 8 : Reprenez et amÉliorez un projet existant

Optimiser les performances de l'application web « SokoNyan » en réduisant le nombre de requêtes http du navigateur et en diminuant le poids des fichiers chargés.

Optimiser les performances

PROJET 8 : REPRENEZ ET AMÉLIOREZ UN PROJET EXISTANT

Table des matières

ÉTAT DES LIEUX

L'application

L'application est fonctionnelle et hébergée sur un serveur. Elle est accessible par internet au lien suivant :

https://trsrv.ddns.net/oc/projet8

Le projet d'origine a été développé de façon modulaire en utilisant **AMD** (Asynchronous Module Defininition) et la bibliothèque **RequireJS**. RequireJS permet d'organiser les modules JS et de gérer leurs dépendances. Chaque module est construit en suivant les règles des modules AMD, qu'ils soient répartis dans plusieurs fichiers ou dans un seul. Il s'agit d'une très bonne pratique de développement qui permet de bien organiser son code de manière durable et facilement soutenable, ce qui augmente les performances en chargeant les modules seulement lorsqu'ils sont nécessaires pour l'application.

Conformément au cahier des charges, le code a été modifié et comprend une nouvelle aventure avec 3 nouveaux niveaux et un PNJ (Personnage Non Joueur) déplaçant en boucle dans un parcours prédéfini pour chaque nouveau niveau de l'aventure.

Cette nouvelle fonctionnalité du code a déjà été optimisée en partie grâce à l'utilisation de **sprite** (plusieurs images regroupées dans un seul fichier d'image). Cette méthode a permis de considérablement diminuer les requêtes http du navigateur web ainsi que le poids des images chargées.

Le serveur

L'application et la base de données sont hébergées sur un serveur distant personnel.

Wamp :

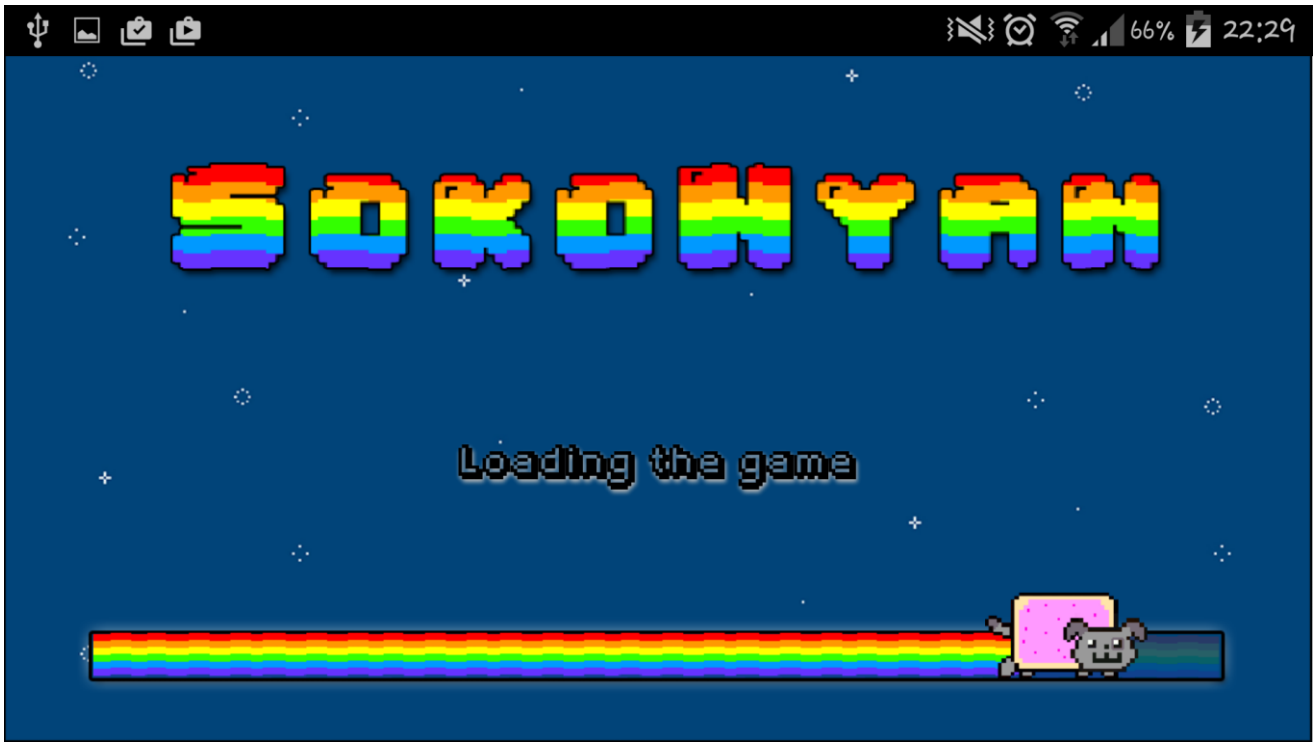
- Apache version 2.4.9
- PHP version 5.5.12
- MySQL version 5.6.17

OS : Windows10

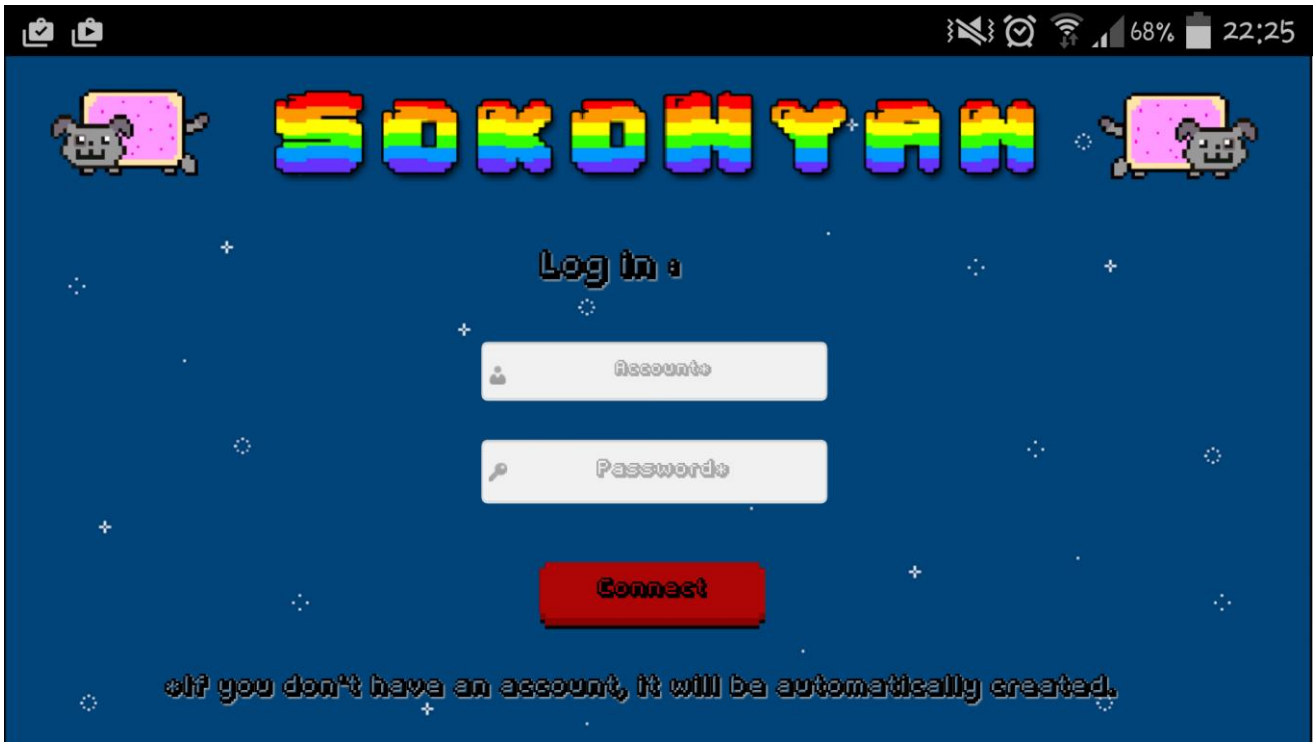
1er lancement de l'application

Jusqu'à ce qu'on arrive à la page d'authentification et grâce à l'outil de développement du navigateur web **FireFox**, nous allons étudier le temps de chargement ainsi que le nombre de requêtes http à l'issue de la phase de pré chargement de l'application.

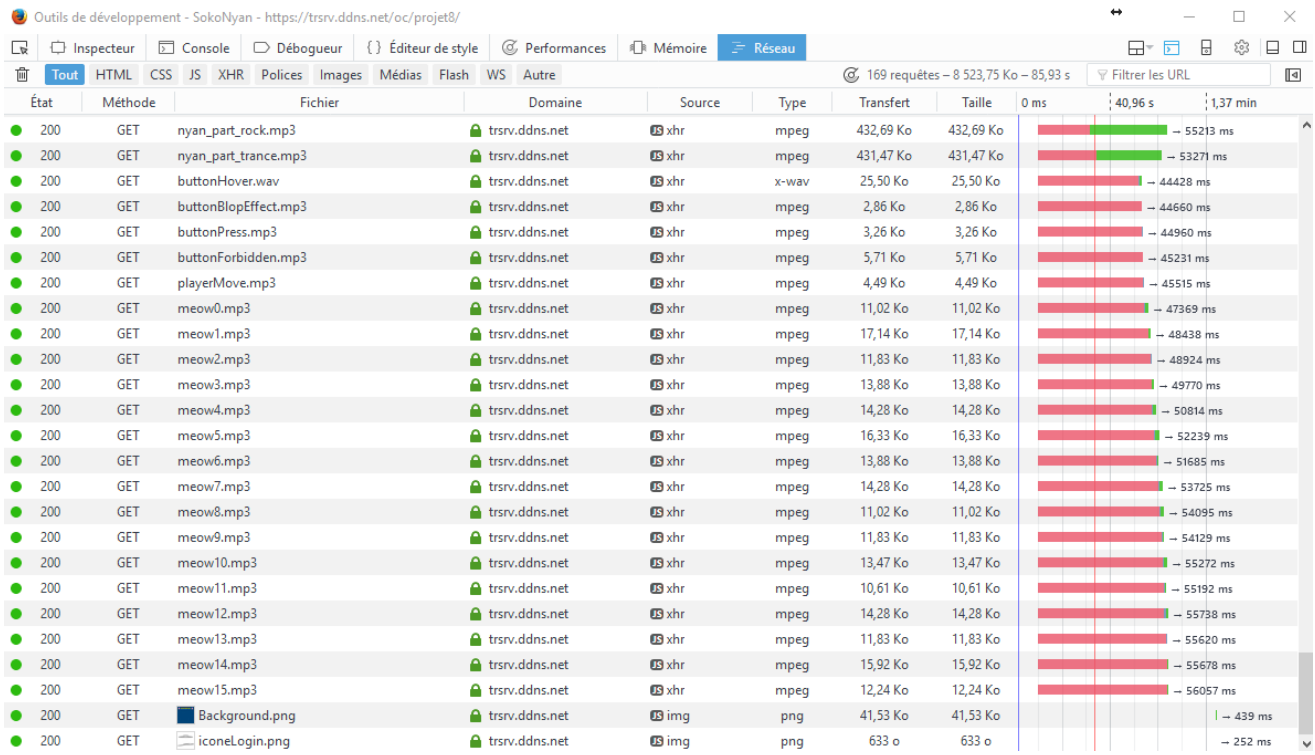
Lancement de l'application : pré chargement des images, sons, fichiers css, js, html, etc…



Chargement terminé : écran d'authentification



Résultats affichés dans l'outil de développement FireFox



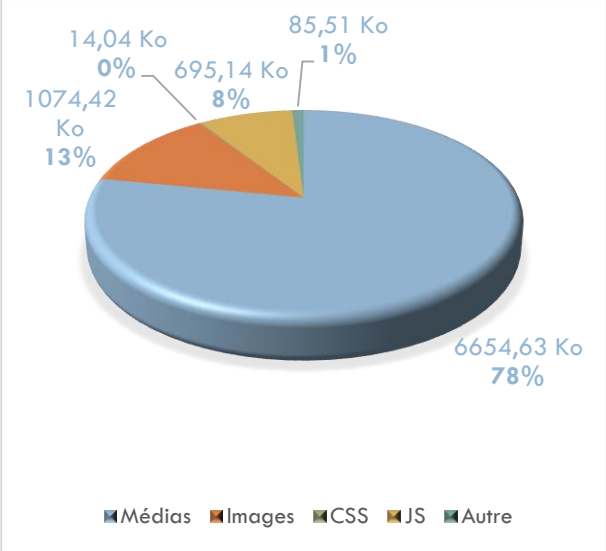
Analyse et diagnostic

Nous constatons que le chargement jusqu'à la page d'authentification :

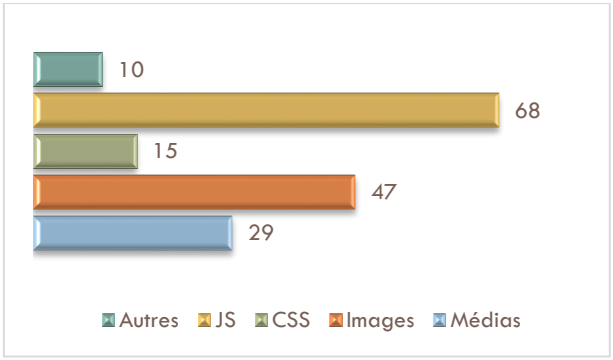
- a pris un délai de **85,93 secondes,**
- a nécessité **169 requêtes http**
- et le navigateur a chargé du contenu pour un total de **8523,75 Ko**

Ces 3 points sont trop élevés. Nous allons découper ces informations par type de chargement.

Répartition des transferts par type de fichier



Nombre de requêtes par type de fichier



Dans un premier temps, notre attention est retenue par la forte proportion de médias (sons et musiques) dans le transfert : soit 78% pour un poids total de 6654.63 Ko et un nombre de requêtes moyen s'élevant à 29. La principale cause semble être le chargement de musiques assez lourdes et nombreuses. À l'inverse, les bruitages sont plus élevés en nombre, mais restent assez légers. Nous allons donc étudier comment réduire ce média voire en retirer certains dans le pré chargement sans que la qualité en pâtisse véritablement.

Les images arrivent en deuxième position en termes de poids 1074,42 Ko : soit 13% du total pour un volume assez élevé de 47 requêtes http. Retenons que le nombre de requêtes et de poids a déjà été optimisé grâce au chargement de feuilles de sprites plutôt que d'image par image. Il est encore possible de réduire le poids en compressant certaines images assez lourdes et en utilisant le CSS plutôt que le chargement d'images.

Les fichiers JS sont les plus nombreux 68 et pèsent 695,14 Ko. Il est possible de réduire complètement le nombre de requêtes et de réduire considérablement leurs poids en les minimisant ou les minifiant à l'aide d'outil adéquat.

Les fichiers CSS représentent une infime partie au niveau du poids et peu de requêtes 15. Il est également possible comme pour les fichiers JS de les réduire complètement en les minifiant.

Les autres fichiers, regroupant les langues, la police, et le fichier HTML, ne seront pas réduits.

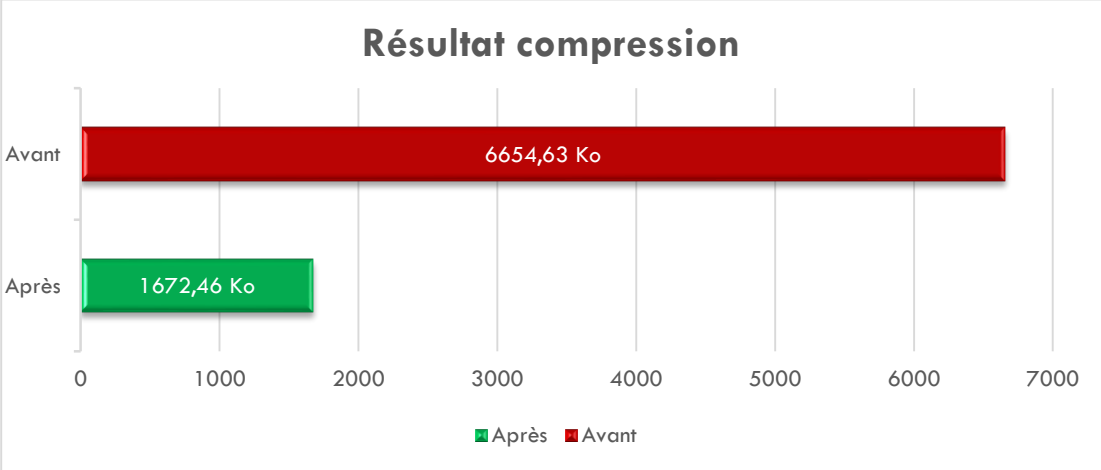
OPTIMISATION DES MÉDIAS

Compressions des médias

Par média nous entendons sons, bruitages et musiques.

Dans un premier temps, nous allons compresser tous les fichiers et les convertir au format MP3. Volontairement nous n'allons pas utiliser d'autre format de fichier tel que OGG. Il aurait été plus intéressant, semble-t-il, d'effectuer un contrôle du format accepté par le navigateur de l'internaute avant le démarrage du pré chargement afin d'uniquement charger le format compatible au lieu des deux. Finalement, le parti pris est d'utiliser un seul format : le MP3.

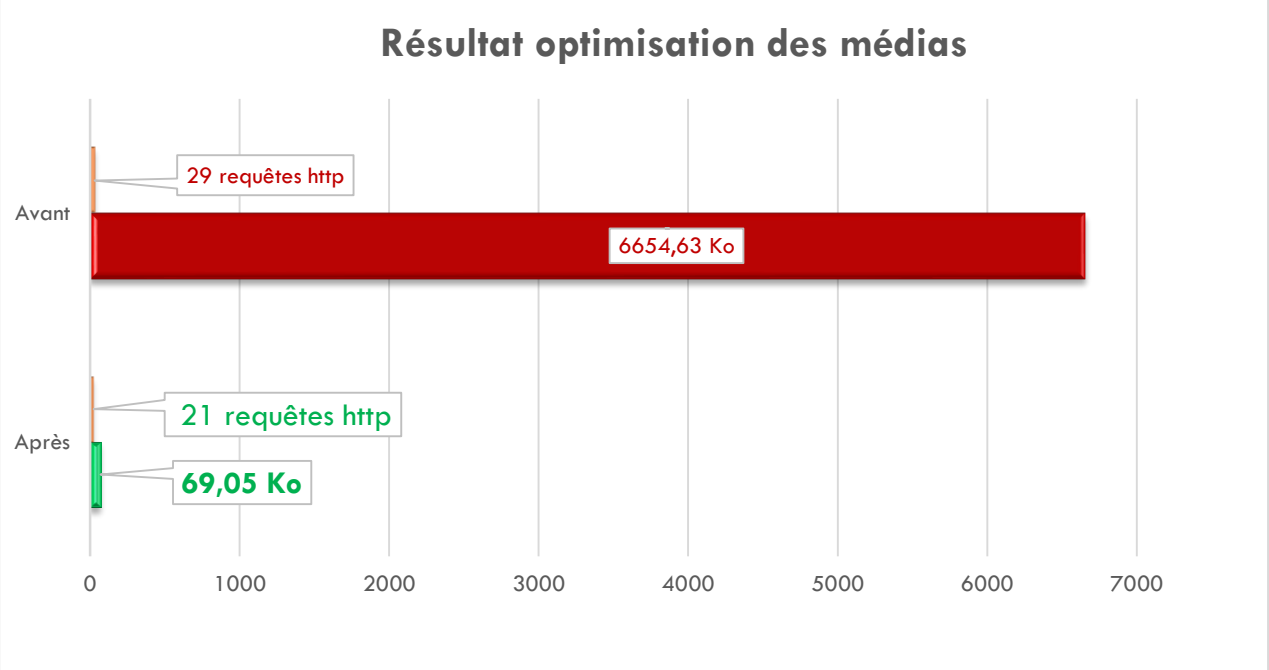
Grâce à la compression, j'ai pu faire baisser le poids de 6654.63 Ko à 1672.46 Ko. Le total est quasi **divisé par 4** et ce, pour une très légère baisse de qualité au niveau sonore.



Retrait des musiques du pré chargement

En ne pré chargeant pas les musiques, il est possible de faire mieux. C'est pourquoi l'application lancera la musique sur demande. Ainsi, on ne charge que les bruitages, ce qui nous ramène à 21 requêtes http et un chargement de médias s'élevant à seulement **69,05 Ko.** On est loin du 6654.63 Ko d'origine et on l'a quasi diminué de 100 fois. Cependant cette optimisation est à relativiser. Il y a certes un gain de temps pour accéder à l'application, mais la navigation risque d'être moins fluide pendant le chargement des musiques entre les différents écrans.

Résultat



OPTIMISATIONS DES IMAGES

Compression des images

Privilégier le format « jpg » lorsqu'il n'y a pas de transparence.

Utiliser le « png » quand il y a de la transparence.

Utiliser les sprites

Autant que possible, il faut regrouper les images de même taille dans une feuille de sprites (sprite sheet). Cela permettra de limiter les requêtes htpp et potentiellement limiter le transfert global.

À noter qu'une bonne partie des images a déjà été regroupée en sprite sheet à l'issue du développement de l'étape n°3 du projet (nouvelle aventure, flags, boutons, etc.)

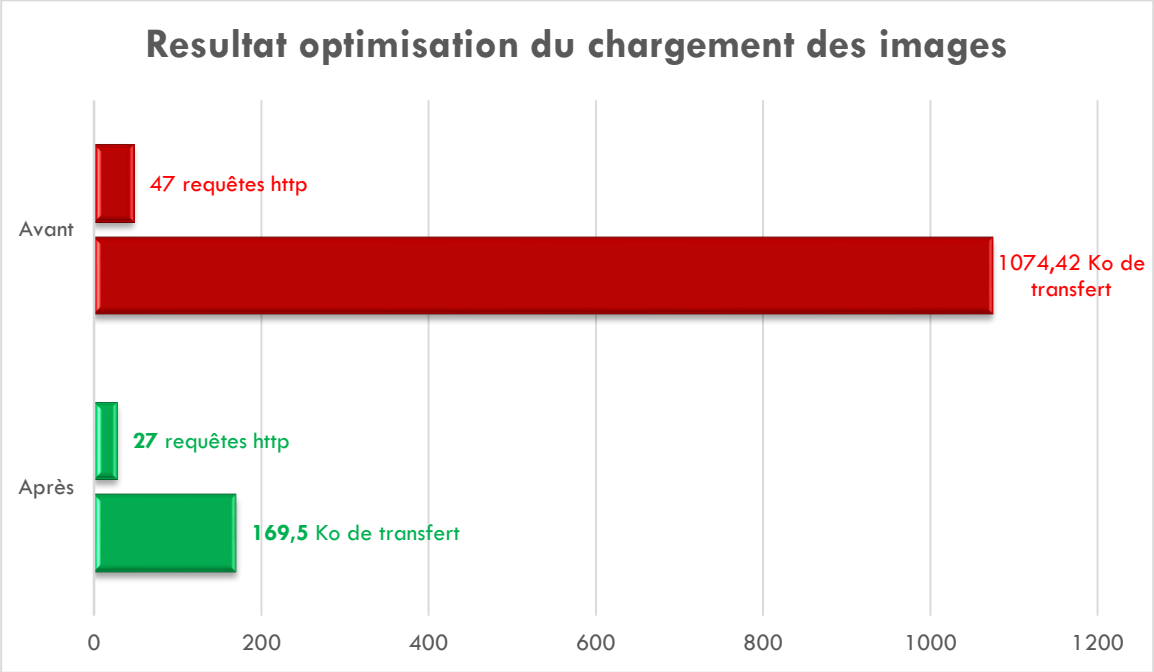
Eviter les doublons

Certains fichiers étaient pré chargés plusieurs fois inutilement.

Privilégié le CSS aux images volumineuses

En effet, plutôt que de charger une image volumineuse qui comprend en réalité plusieurs images, il est plus judicieux de découper les petites images composant l'image volumineuse en plusieurs sprites et les réutiliser par la suite. Ainsi nous évitons le chargement de plusieurs images de background tel que le menu, l'aide, l'écran de login et l'écran de pré chargement.

Résultat



Le poids des images a été diminué de plus de 6 fois pour un total de **169,5 Ko** : soit seulement 15% du poids de départ. Pour un nombre de requêtes http optimisé à **27 requêtes** soit 57.45 % du nombre de requêtes de départ.

OPTIMISATION DES FICHIERS JS ET CSS

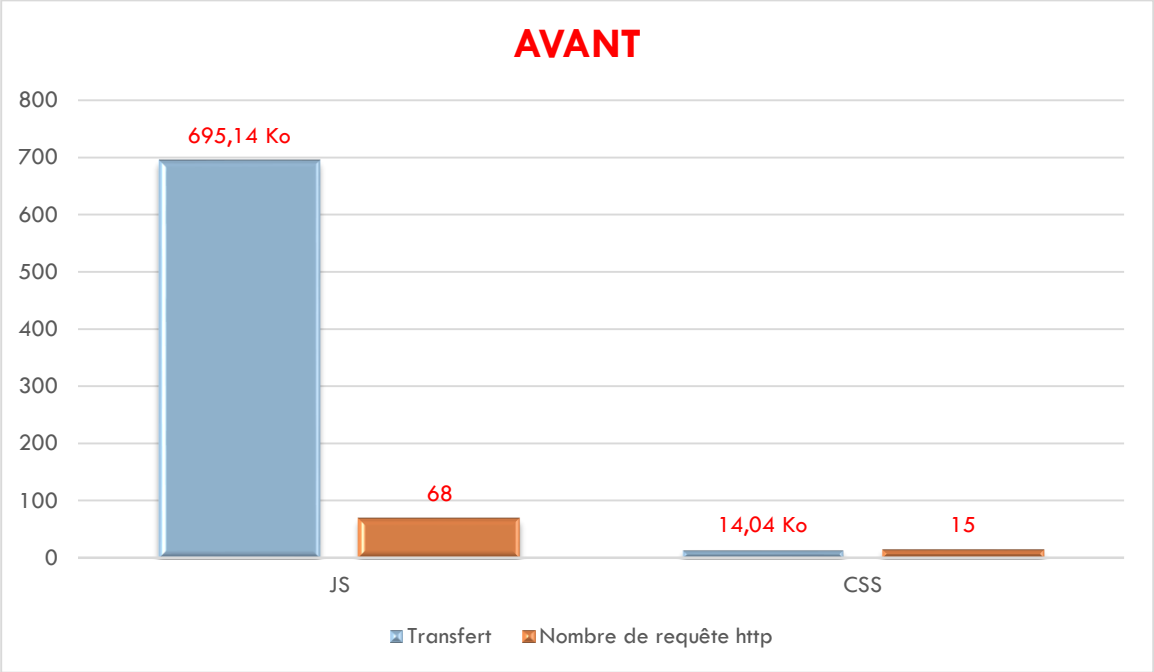
Regrouper et minifier les fichiers JS et CSS

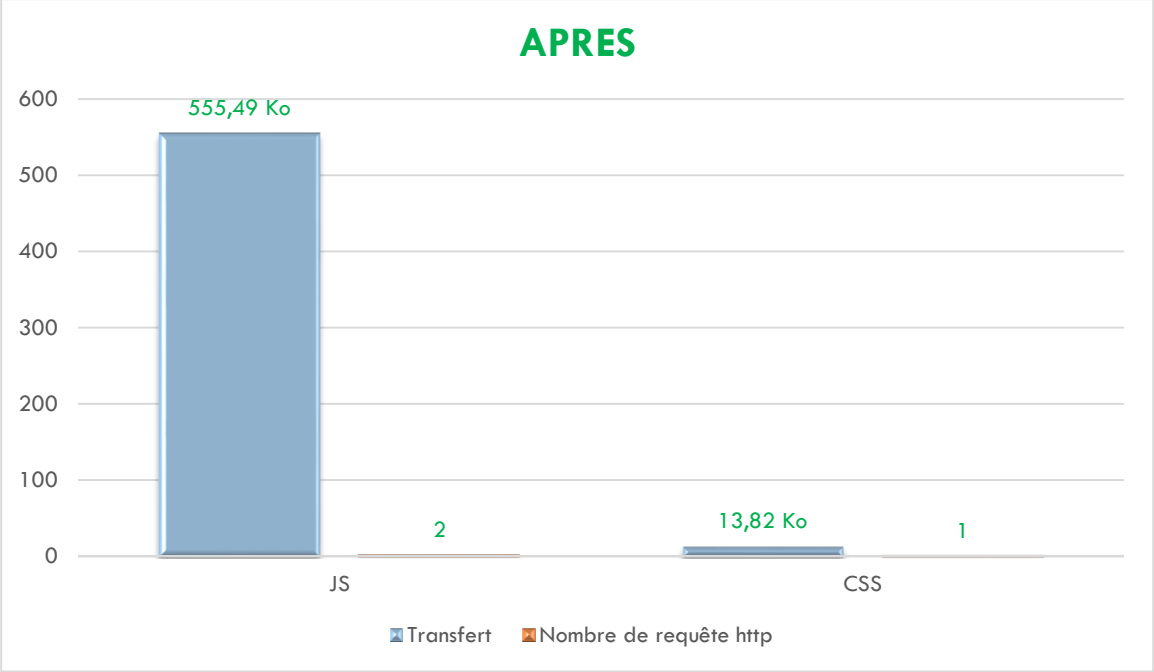
En suivant les instructions du site http://requirejs.org/docs/optimization.html

Il est possible de minifier la totalité des fichiers JS et CSS en un seul.

Cela réduira le poids des fichiers (suppression des espaces et commentaires) et surtout le nombre de requêtes http sera réduit.

Résultat

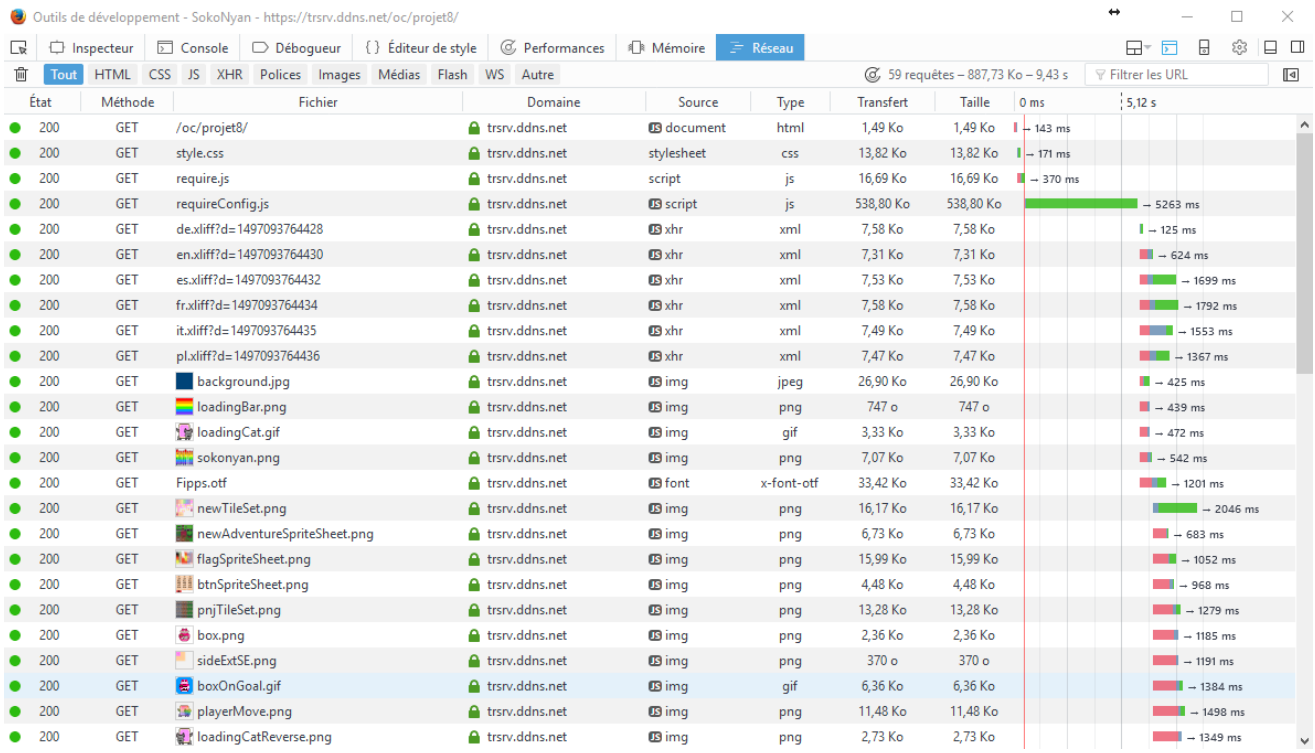




Le poids des fichiers JS a été réduit de plus de 20% pour le CSS. La réduction est légère, à peu près de 2%. Mais pour les deux types de fichiers, la réduction la plus significative est au niveau du nombre de requêtes Htpp :

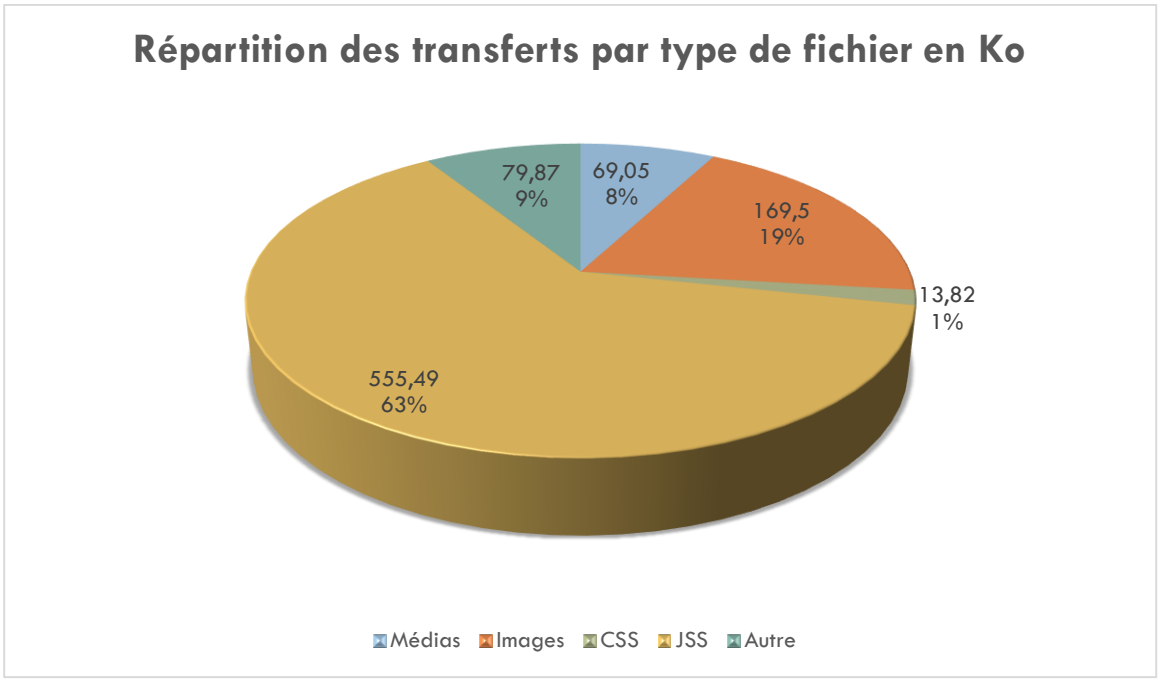
- 97 % pour le JS soit un nombre de 2 requêtes, au départ il y en avait 68.
- 93 % pour le CSS soit 1 requête, au départ il y en avait 15.

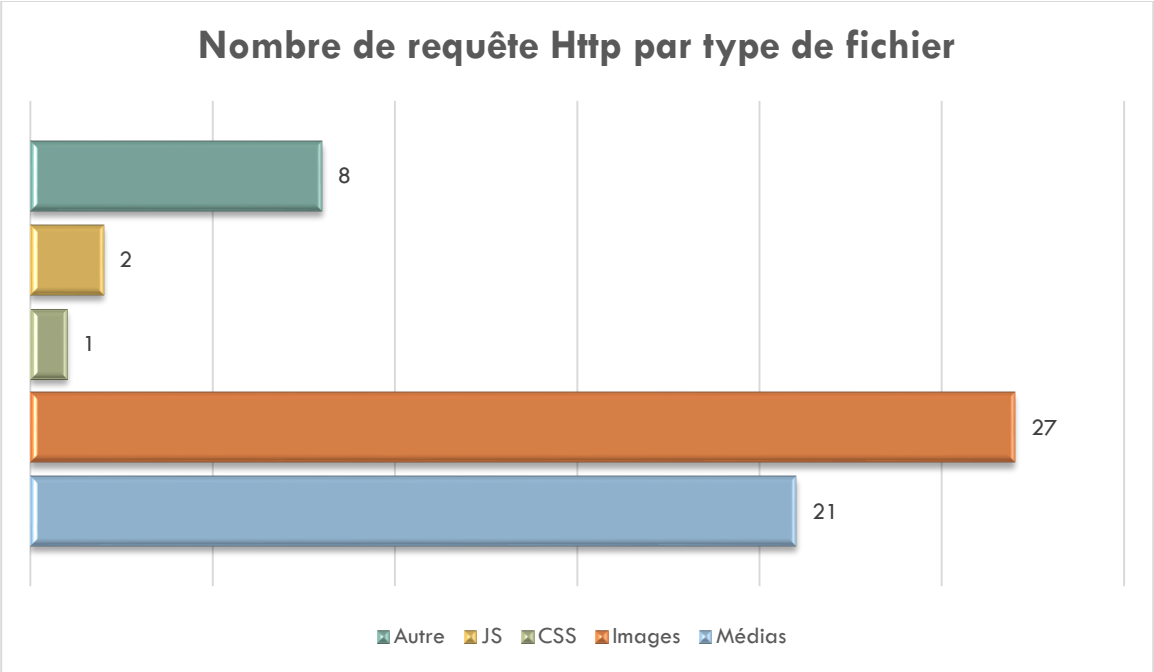
RESULTAT FINAL APRES OPTIMISATION



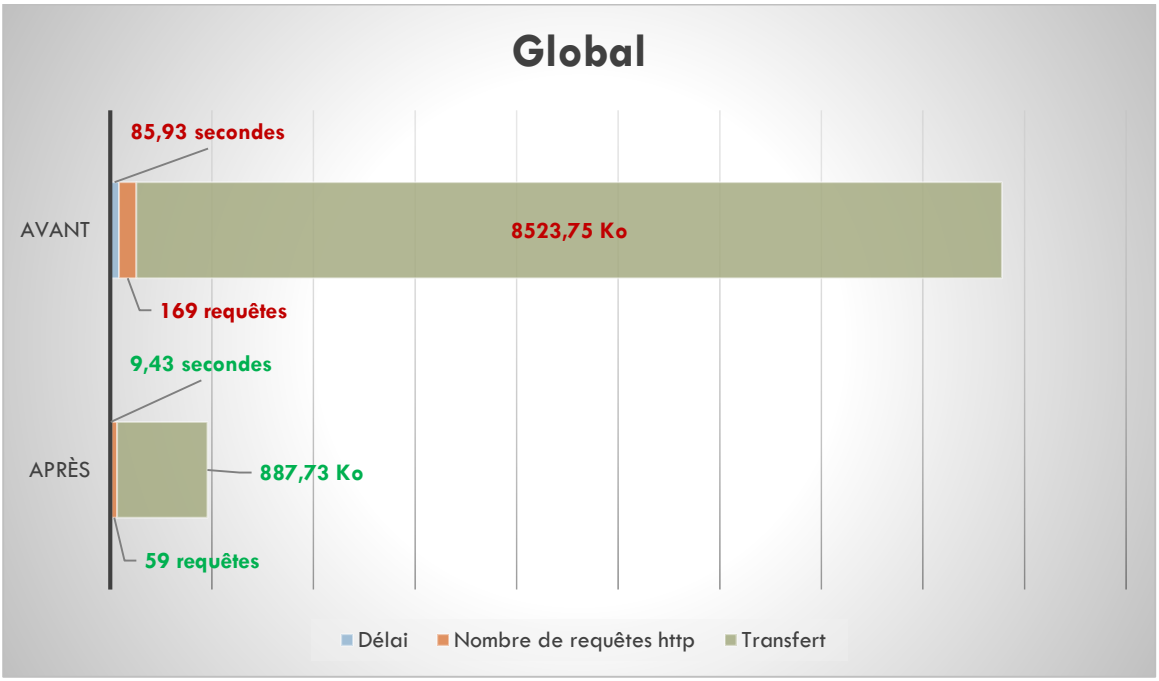
État	Méthode	Fichier	Domaine	Source	Type	Transfert	Taille	0 ms / 5,12 s
200	GET	/oc/projet8/	trsrv.ddns.net	document	html	1,49 Ko	1,49 Ko	143 ms
200	GET	style.css	trsrv.ddns.net	stylesheet	css	13,82 Ko	13,82 Ko	171 ms
200	GET	require.js	trsrv.ddns.net	script	js	16,69 Ko	16,69 Ko	370 ms
200	GET	requireConfig.js	trsrv.ddns.net	script	js	538,80 Ko	538,80 Ko	5263 ms
200	GET	de.xliff?d=1497093764428	trsrv.ddns.net	xhr	xml	7,58 Ko	7,58 Ko	125 ms
200	GET	en.xliff?d=1497093764430	trsrv.ddns.net	xhr	xml	7,31 Ko	7,31 Ko	624 ms
200	GET	es.xliff?d=1497093764432	trsrv.ddns.net	xhr	xml	7,53 Ko	7,53 Ko	1699 ms
200	GET	fr.xliff?d=1497093764434	trsrv.ddns.net	xhr	xml	7,58 Ko	7,58 Ko	1792 ms
200	GET	it.xliff?d=1497093764435	trsrv.ddns.net	xhr	xml	7,49 Ko	7,49 Ko	1553 ms
200	GET	pl.xliff?d=1497093764436	trsrv.ddns.net	xhr	xml	7,47 Ko	7,47 Ko	1367 ms
200	GET	background.jpg	trsrv.ddns.net	img	jpeg	26,90 Ko	26,90 Ko	425 ms
200	GET	loadingBar.png	trsrv.ddns.net	img	png	747 o	747 o	439 ms
200	GET	loadingCat.gif	trsrv.ddns.net	img	gif	3,33 Ko	3,33 Ko	472 ms
200	GET	sokonyan.png	trsrv.ddns.net	img	png	7,07 Ko	7,07 Ko	542 ms
200	GET	Fipps.otf	trsrv.ddns.net	font	x-font-otf	33,42 Ko	33,42 Ko	1201 ms
200	GET	newTileSet.png	trsrv.ddns.net	img	png	16,17 Ko	16,17 Ko	2046 ms
200	GET	newAdventureSpriteSheet.png	trsrv.ddns.net	img	png	6,73 Ko	6,73 Ko	683 ms
200	GET	flagSpriteSheet.png	trsrv.ddns.net	img	png	15,99 Ko	15,99 Ko	1052 ms
200	GET	btnSpriteSheet.png	trsrv.ddns.net	img	png	4,48 Ko	4,48 Ko	968 ms
200	GET	pnjTileSet.png	trsrv.ddns.net	img	png	13,28 Ko	13,28 Ko	1279 ms
200	GET	box.png	trsrv.ddns.net	img	png	2,36 Ko	2,36 Ko	1185 ms
200	GET	sideExtSE.png	trsrv.ddns.net	img	png	370 o	370 o	1191 ms
200	GET	boxOnGoal.gif	trsrv.ddns.net	img	gif	6,36 Ko	6,36 Ko	1384 ms
200	GET	playerMove.png	trsrv.ddns.net	img	png	11,48 Ko	11,48 Ko	1498 ms
200	GET	loadingCatReverse.png	trsrv.ddns.net	img	png	2,73 Ko	2,73 Ko	1349 ms

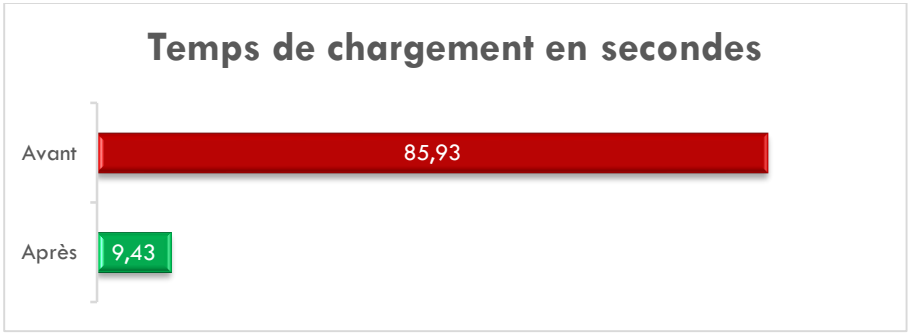
- Un temps de chargement de **9.43 secondes**
- Le nombre de requêtes http s'élève à **59**
- Le contenu chargé s'élève à **887.73 Ko**




Nombre de requête Http par type de fichier
8
2
1
27
21
Autre
JS
CSS
Images
Médias

Comparatif


Global
85,93 secondes
AVANT
8523,75 Ko
169 requêtes
9,43 secondes
APRÈS
887,73 Ko
59 requêtes
Délai
Nombre de requêtes http
Transfert



Le temps de chargement est fortement amélioré (**plus de 9 fois plus rapide que le chargement d'origine**) et représente 10.97% du temps de chargement d'origine.

Le nombre de requêtes http d'origine est diminué de 65.01 % et **représente le tiers des requêtes d'origine.**

Enfin**, le poids du chargement global d'origine a été diminué de plus de 9 fois :** soit 10.41% du chargement de départ.