

# Webpack

Matthieu Nicolas  
Licence Pro CIASIE

# Objectifs

























- Vous sensibiliser aux problématiques de performances et de mise en prod
- Vous faire découvrir plusieurs de ces problématiques et les solutions actuelles
- Vous présenter un des outils existants pour y répondre

# Performances

- Applications de plus en plus conséquentes
  - + en + de fichiers
  - + en + de code
- Cela impacte négativement les performances
  - Et donc l'expérience utilisateur

# Trop de fichiers...

- Une requête HTTP est nécessaire pour chaque fichier

Status	Method	File	Domain	Cause	Type	Transferred	Size	0 ms	160 ms	320 ms	480 ms
200	GET	 index.js	 localhost:8080	script	js	2.08 KB	1.78 KB		 10 ms		
200	GET	 Album.js	 localhost:8080	script	js	564 B	260 B		 4 ms		
200	GET	 Game.js	 localhost:8080	script	js	594 B	290 B		 4 ms		
200	GET	 Movie.js	 localhost:8080	script	js	634 B	330 B		 4 ms		
200	GET	 App.js	 localhost:8080	script	js	8.45 KB	8.15 KB		 4 ms		
200	GET	 css?family=Lato:400,700,400itali...	 fonts.googleapis.com	stylesheet	css	948 B	3.02 KB		 18 ms		
200	GET	 Media.js	 localhost:8080	script	js	503 B	199 B			 58 ms	
200	GET	 Collection.js	 localhost:8080	script	js	722 B	418 B			 58 ms	

# Latence réseau

- Tout le monde n'a pas le serveur qui tourne en local sur sa machine
- Chaque requête implique une latence réseau
  - 30-50ms pour les requêtes continentales
  - 90ms pour chaque requête transatlantique
- Ces latences réseaux s'additionnent

# Bundle

- Pour limiter ce problème, on va réduire le nombre de fichiers
- On va regrouper tous les fichiers ensemble dans un **bundle**

Status	Method	File	Domain	Cause	Type	Transferred	Size	0 ms	160 ms	320 ms	480 ms	640 ms	800 ms ▲
200	GET	jquery-3.3.1.js	localhost:8080	script	js	265.68 KB	265.38 KB	11 ms					
200	GET	semantic.min.js	localhost:8080	script	js	269.57 KB	269.27 KB	10 ms					
200	GET	bundle.js	localhost:8080	script	js	8.26 KB	7.97 KB	3 ms					
200	GET	css?family=Lato:400,700,400itali...	fonts.googleapis.com	stylesheet	css	948 B	3.02 KB		164 ms				
200	GET	S6u9w4BMUTPHh6UVSwiPGQ3...	fonts.gstatic.com	font	woff2	14.21 KB	13.75 KB			175 ms			
200	GET	S6uyw4BMUTPHjx4wXiWtFCc.w...	fonts.gstatic.com	font	woff2	14.08 KB	13.62 KB			208 ms			

# HTTP/2

- La nouvelle version du protocole HTTP, **HTTP/2**, a notamment pour but de répondre à ce problème
- Permet d'effectuer plusieurs requêtes sur une même connexion TCP
- Le jour où **HTTP/2** sera répandu, moins d'intérêt à **bundler**

# Trop de code...

- *Le code est écrit pour être lu*
  - Plus précisément, lu par des humains
- Cela se répercute sur son contenu
  - Noms de variable *explicites*
  - Code *aéré, indenté*



# Bande passante

- L'ordinateur n'a pas besoin de tout ces octets en trop...

```
1  class Media {  
2  
3  constructor(title, releaseDate, rating, img) {  
4      this.title = title  
5      this.releaseDate = releaseDate  
6      this.rating = rating  
7      this.img = img  
8  }  
9  }  
10
```

- ... et nous, on a pas tous la fibre

# Minification

- Peut supprimer les caractères inutiles, raccourcir les noms de variables...
- On appelle ce processus **minification** du code (30-40% de gain)

```
1 class e{constructor(t,i,n,e){this.title=t,this.releaseDate=i,this.rating=n,this.img=e}}
```

# Outils

- Différents outils permettent d'effectuer ces tâches
  - Grunt
  - Rollup
  - **Webpack**

# Webpack

- Par défaut, permet de packager notre application...
  - Prend un fichier d'entrée dans l'application
  - Établit le graphe des dépendances de notre programme
  - Minifie et concatène le code dans un fichier de sortie
- ... mais pas que

# Tree shaking

- En étudiant les **import/export** de notre application, capable de déterminer le code *mort*
  - Par exemple une fonction exportée qui n'est jamais importée
- **Webpack** va retirer ce code inutile du **bundle** final

# Installation

- S'installe facilement avec **npm**
  - Voir **nvm** si **npm** n'est pas installé sur votre machine
- À la racine de votre dossier
  - *npm install -- save-dev webpack*
  - *npm install -- save-dev webpack-cli*
- Installe **Webpack** dans le dossier *node\_modules/.bin/webpack*

# Configuration

- Ajouter à la racine du projet le fichier *webpack.config.js*

```
1  const path = require("path")
2
3  module.exports = {
4    entry: "./src/index.js",
5    output: {
6      path: path.resolve(__dirname, "dist"),
7      filename: "bundle.js"
8    },
9    mode: "production"
10 }
11 |
```

Exemple de *webpack.config.js*

# Utilisation

- Ajouter dans *package.json* un script *build*

```
10  .. "scripts": {  
11  |   .. "build": "webpack"  
12  .. },
```

- À la racine de votre projet
  - *npm run build*
- Génère le fichier résultat à la sortie spécifiée dans la configuration
- Reste plus qu'à modifier l'HTML pour utiliser le bundle



# Conclusion

- Il est nécessaire d'adopter des bonnes pratiques de packaging pour les applis web
- De nombreux outils existent, choisissez-en un
- Optimiser votre algorithme, c'est bien...
- ... mais optimiser votre build, c'est aussi important

# Références Webpack

- Concepts
  - <https://webpack.js.org/concepts>
- Guides
  - <https://webpack.js.org/guides/>

# Pour aller plus loin

- Web Performance 101
  - <https://3perf.com/talks/web-perf-101/>
- Google PageSpeed Insights
  - <https://developers.google.com/speed/pagespeed/insights/>

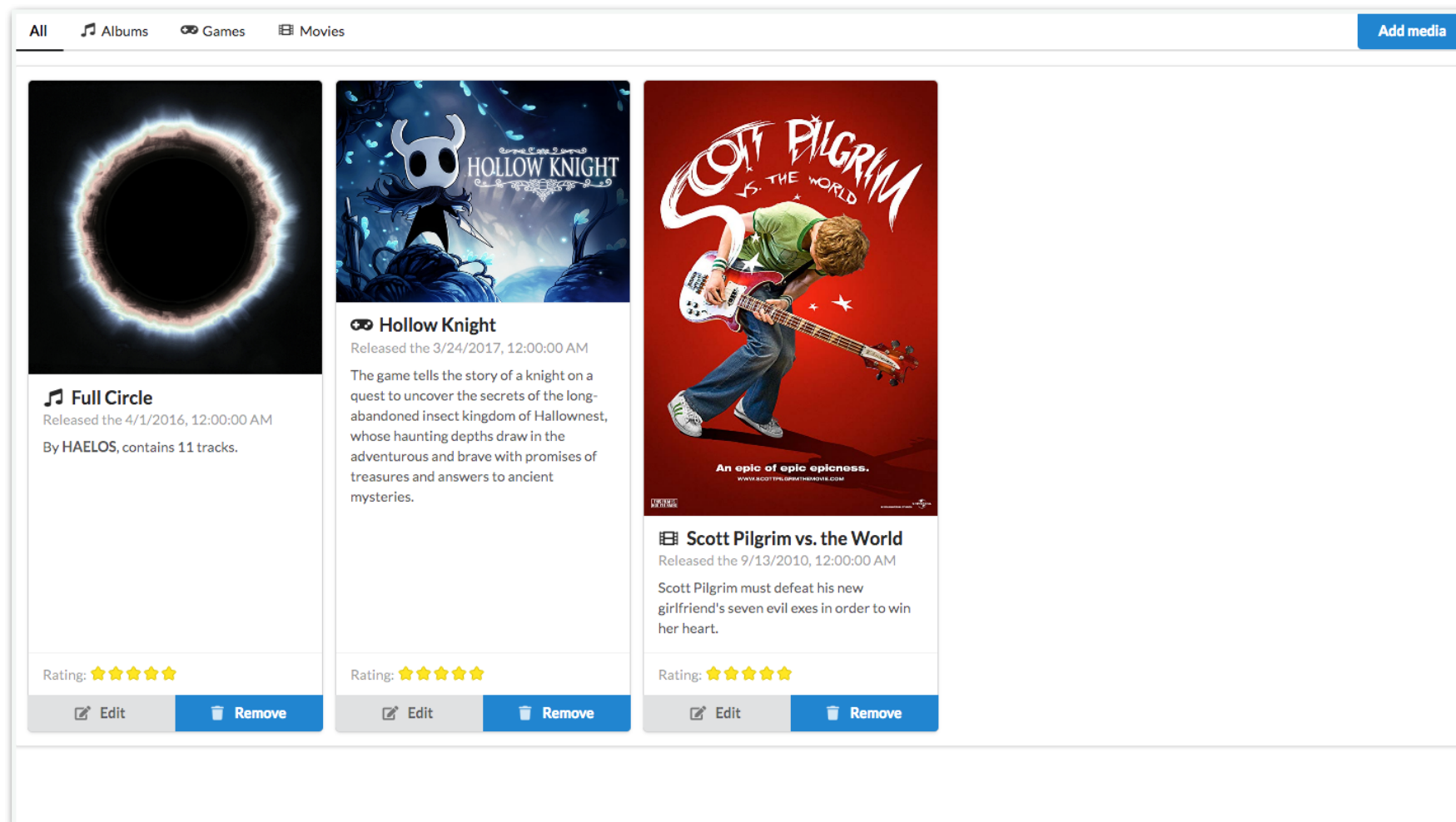
# Overture

- State of JS 2018
  - <https://2018.stateofjs.com/>

TP

# Consignes

- On continue sur l'application de médiathèque



- On utilise **Webpack** pour la packager

# Let's go

Toujours sur le même repo