Inclure du javascript de manière performante

Jean-pierre VINCENT
Braincracking.org

Orateur

Jean-pierre VINCENT

En ligne :

- **braincracking**.org (veille)
- twitter.com/**theystolemynick**
- jp@vincent.net
- google it : jpvincent

10 ans de Web dont :

- 3 ans de Yahoo!
- 2 ans d'application Web (timeofmylife.com)

Pourquoi ?

rendu rapide = argent

bit.ly/perfs_benefice
bit.ly/perfs_benefice2

Comment

Inclure JS de manière non bloquante

(entre autres)

Le fautif

```
<script
src="my.js"></script>
</head>
```

Le fautif

```
<script
src="my.js"></script>
</head>
```

- bloque ce qui vient après :
  - rendu

Le fautif

```
<script
src="my.js"></script>
</head>
```

- bloque ce qui vient après :
  - rendu
  - exécution

Le fautif

```
<script
src="my.js"></script>
</head>
```

- bloque ce qui vient après :
  - rendu
  - exécution
  - téléchargement

Le fautif

```
<script
src="my.js"></script>
</head>
```

- bloque ce qui vient après :
  - rendu
  - exécution
  - téléchargement
- IE6-7 : un seul JS à la fois

exemple

réservation voyages-sncf.com :

- HTML obtenu en 200ms (!)

exemple

réservation voyages-sncf.com :

- HTML obtenu en 200ms (!)
- certains CSS arrivent en 50ms (!!)

(je voudrais bien jouer avec ce ping là)

exemple

réservation voyages-sncf.com :

- HTML obtenu en 200ms (!)
- certains CSS arrivent en 50ms (!!)

(je voudrais bien jouer avec ce ping là)

- au final : 2s de page blanche

#fail

bit.ly/perfs_sncf

Objectif

remplacer

```
<script
src="my.js"></script>
</head>
```

Alternatives

- Inline

Alternatives

- Inline
- Bottom

Alternatives

- Inline
- Bottom
- DOM

Alternatives

que des désavantages

Alternatives

que des désavantages

mais

Alternatives

que des désavantages
mais

non bloquant

(= argent)

Inline

Inline

```
<script>
 <?php include('my.js'); ?>
</script>
</head>
```

Inline

- ✓ ordre d'exécution

Inline

- ✓ ordre d'exécution
- ✓ dépendances inline

Inline

- ✓ ordre d'exécution
- ✓ dépendances inline
- ✓ non bloquant (ou presque : bit.ly/inline_bloquant)

Inline

- ✓ ordre d'exécution
- ✓ dépendances inline
- ✓ non bloquant (ou presque : bit.ly/inline_bloquant)
- ✓ document.write()

Inline

- ✓ ordre d'exécution
- ✓ dépendances inline
- ✓ non bloquant (ou presque : bit.ly/inline_bloquant)
- ✓ document.write()
- ✓ pas de HTTP

Inline

- ✓ ordre d'exécution
- ✓ dépendances inline
- ✓ non bloquant (ou presque : bit.ly/inline_bloquant)
- ✓ document.write()
- ✓ pas de HTTP

- ✗ pas de cache

Inline

Bon pour :

- beaucoup de nouveaux visiteurs

Inline

Bon pour :

- beaucoup de nouveaux visiteurs
- peu de JS

Inline

Bon pour :

- beaucoup de nouveaux visiteurs
- peu de JS
- bonne bande passante (France)

Inline

- évènementielle (jeux-concours)

Inline

- évènementielle (jeux-concours)

- Homepage, sans cookie
  - Google Search, Yahoo! (avant)
  - Netvibes, Facebook

Inline

- évènementielle (jeux-concours)

- Homepage, sans cookie
  - Google Search, Yahoo! (avant)
  - Netvibes, Facebook

- pages d'entrées
  - login
  - signup

Bottom

```
<script
src="my.js"></script>
</body>
```

Bottom

Bottom

- ✓ ordre d'exécution

Bottom

- ✓ ordre d'exécution
- ✓ non bloquant

Bottom

- ✓ ordre d'exécution
- ✓ non bloquant
- ✓ mise en cache

Bottom

- ✓ ordre d'exécution
- ✓ non bloquant
- ✓ mise en cache

- ✗ dépendances inline

Bottom

✓ ordre d'exécution
✓ non bloquant
✓ mise en cache

✗ dépendances inline
✗ document.write()

Bottom

Bon pour les pages dont :

Bottom

Bon pour les pages dont :

- le contenu textuel est primordial

Bottom

Bon pour les pages dont :

- le contenu textuel est primordial
- le HTML est rapide

Bottom

Bon pour les pages dont :

- le contenu textuel est primordial
- le HTML est rapide
- reçoit des visiteurs zappeurs

Bottom

Bon pour les pages dont :

- le contenu textuel est primordial
- le HTML est rapide
- reçoit des visiteurs zappeurs
- JS est secondaire mais lourd
  - publicités
  - widgets

Bottom

Type de pages :

- Blogs, Journaux

Bottom

Type de pages :

- Blogs, Journaux
- Boutiques

Bottom

Type de pages :

- Blogs, Journaux
- Boutiques
- Comparateurs

Bottom

Type de pages :

- Blogs, Journaux
- Boutiques
- Comparateurs
- …

Bottom - condition 1

contenu principal envoyé
en moins de 500ms ?

Bottom - condition 1

contenu principal envoyé
en moins de 500ms ?

- flush()

Bottom - condition 1

contenu principal envoyé
en moins de 500ms ?

- flush()
- cache serveur

Bottom - condition 1

contenu principal envoyé
en moins de 500ms ?

- flush()
- cache serveur
- en premier dans la source

Bottom - condition 2

dépendances JS inline
correctibles ?

Bottom - condition 2

dépendances JS inline
correctibles ?

- mute + eval() : bit.ly/mute_eval

Bottom - condition 2

dépendances JS inline correctibles ?

- mute + eval() : bit.ly/mute_eval
- hijacking

Bottom - condition 2

dépendances JS inline
correctibles ?

- mute + eval() : bit.ly/mute_eval
- hijacking

Si non, modification du JS obligatoire

DOM

DOM

```
function loader(fCallback) {




}
```

DOM

```
function loader(fCallback) {
var oScript =
document
.getElementsByTagName("head")[0]
 .createElement('script');



}
```

DOM

```
function loader(fCallback) {
var oScript =
document
.getElementsByTagName("head")[0]
 .createElement('script');
oScript.src= 'my.js';



}
```

DOM

```
function loader(fCallback) {
var oScript =
document
.getElementsByTagName("head")[0]
 .createElement('script');
oScript.src= 'my.js';
oScript.onload = fCallback;
oScript.onreadystatechange =
fCallback;
}
```

DOM

jQuery :

```
$.getScript(
   'my.js',
   fCallback);
```

YUI 3 :

```
Y.Get.script(
   'my.js',
    {onSuccess: fCallback} );
```

DOM

- ✓ non bloquant

DOM

- ✓ non bloquant
- ✓ mise en cache

DOM

- ✓ non bloquant
- ✓ mise en cache
- ✓ IE6-7 parallélise ! (bit.ly/dom_IE_para)

DOM

- ✓ non bloquant
- ✓ mise en cache
- ✓ IE6-7 parallélise ! (bit.ly/dom_IE_para)

- ✗ dépendances inline

DOM

- ✓ non bloquant
- ✓ mise en cache
- ✓ IE6-7 parallélise ! (bit.ly/dom_IE_para)

- ✗ dépendances inline
- ✗ document.write()

DOM

- ✓ non bloquant
- ✓ mise en cache
- ✓ IE6-7 parallélise ! (bit.ly/dom_IE_para)

- ✗ dépendances inline
- ✗ document.write()
- ✗ ordre d'exécution

DOM

Bon pour les pages :

DOM

Bon pour les pages :

- éligibles à Bottom

DOM

Bon pour les pages :

- éligibles à Bottom
- HTML > 0.5 - 1s

ou

DOM

Bon pour les pages :

- éligibles à Bottom
- HTML > 0.5 - 1s

ou

- gros volume de JS (> 500k décompressé)

DOM

Type de pages :

- Blogs, Journaux
- Boutiques
- Comparateurs
- …

DOM

Type de pages :

- Blogs, Journaux
- Boutiques
- Comparateurs
- ...
- **Applications Web**

DOM - préalable

Découper son JS en modules

DOM - préalable

Découper son JS en modules

1 module PHP = 1 fonctionalité = 1 JS

lui donner un identifiant

DOM - préalable

✓ Découper son JS en modules

Centraliser les inclusions :

```
MY.loader =
    function(sScriptID, fCallback) {



};
```

DOM - préalable

✓ Découper son JS en modules

Centraliser les inclusions :

```
MY.loader =
    function(sScriptID, fCallback) {
      $.getScript(
        sScriptID+'.js',
        fCallback);
};
```

DOM - préalable

✓ Découper son JS en modules
✓ Centraliser les inclusions
Corriger toutes ses dépendances :

```
<form id="login-form">
...
</form>
<
```

DOM - préalable

✓ Découper son JS en modules
✓ Centraliser les inclusions
Corriger toutes ses dépendances :

```
<form id="login-form">
...
</form>
<script>

  MY.enrichForm('login-form');

</script>
```

DOM - préalable

✓ Découper son JS en modules
✓ Centraliser les inclusions
Corriger toutes ses dépendances :

```
<form id="login-form">
...
</form>
<script>
MY.loader('login-js', function() {
  MY.enrichForm('login-form');
});
</script>
```

DOM - préalable

- ✓ Découper son JS en modules
- ✓ Centraliser les inclusions
- ✓ Corriger toutes ses dépendances

DOM - préalable

- ✓ Découper son JS en modules
- ✓ Centraliser les inclusions
- ✓ Corriger toutes ses dépendances
- ✓ Concaténer les fichiers communs

DOM - évolution

✓ Découper son JS en modules
✓ Centraliser les inclusions
✓ Corriger toutes ses dépendances
✓ Concaténer les fichiers communs

Plus tard :

- gérer les inclusions multiples

DOM - évolution

✓ Découper son JS en modules
✓ Centraliser les inclusions
✓ Corriger toutes ses dépendances
✓ Concaténer les fichiers communs

Plus tard :

- gérer les inclusions multiples

Tactique YUI3 :

- gérer dépendances entre classes

DOM - évolution

✓ Découper son JS en modules
✓ Centraliser les inclusions
✓ Corriger toutes ses dépendances
✓ Concaténer les fichiers communs

Plus tard :

- gérer les inclusions multiples

Tactique YUI3 :

- gérer dépendances entre classes
- générer les JS à la volée

Etude de cas

Netvibes :

- inline : 100Ko de JS
- DOM : 260Ko de JS
- DOM : widget JS

Etude de cas

Facebook (Wall) :

Etude de cas

Facebook (Wall) :

- flush() HTML presque vide

Etude de cas

Facebook (Wall) :

- flush() HTML presque vide
- inline : 1 loader DOM + 1 loader spécial (bigPipe : bit.ly/fb_bigpipe)

Etude de cas

Facebook (Wall) :

- flush() HTML presque vide
- inline : 1 loader DOM + 1 loader spécial (bigPipe : bit.ly/fb_bigpipe)
- flush() des modules 1 par 1

Etude de cas

Facebook (Wall) :

- flush() HTML presque vide
- inline : 1 loader DOM + 1 loader spécial (bigPipe : bit.ly/fb_bigpipe)
- flush() des modules 1 par 1
- DOM :9 JS chargés (1 principal + 8 modules)

Etude de cas

Facebook (Wall) :

- flush() HTML presque vide
- inline : 1 loader DOM + 1 loader spécial (bigPipe : bit.ly/fb_bigpipe)
- flush() des modules 1 par 1
- DOM :9 JS chargés (1 principal + 8 modules)
- meme technique pour CSS

Etude de cas

YUI 3 :

Etude de cas

YUI 3 :

- tout appel de code commence dans une fonction de callback

Etude de cas

YUI 3 :

- tout appel de code commence dans une fonction de callback
- liste manuelle des dépendances

Etude de cas

YUI 3 :

- tout appel de code commence dans une fonction de callback
- liste manuelle des dépendances
- le loader connait les dépendances et les fichiers déjà chargés

Etude de cas

YUI 3 :

- tout appel de code commence dans une fonction de callback
- liste manuelle des dépendances
- le loader connait les dépendances et les fichiers déjà chargés
- coté serveur : un combinateur

Orateur

Jean-pierre VINCENT

En ligne :

- **braincracking**.org (veille)
- twitter.com/**theystolemynick**
- jp@vincent.net
- google it : jpvincent

10 ans de Web dont :

- 3 ans de Yahoo!
- 2 ans d'application Web (timeofmylife.com)