


# Loader AMD

Pour un code javascript  
performant et organisé

Emmanuel REMY

27/06/2012

# Qui suis je ?

- Un certains nombre d'années dans les technos Web
- Consultant chez Ippon Technologies 
- Rédacteur occasionnel sur *developpez.com*
- Cette présentation et les sources :
  - [https://github.com/emmanuelremy/blog\\_amd](https://github.com/emmanuelremy/blog_amd)



# AMD : Kezako ?

- AMD : Asynchronous Module Definition
- API = indépendance vis à vis du framework
- Adopté par jQuery (1.7), Dojo, Mootools(2), Backbone-aura, Raphael, EmbedJS, Coffee Script (via require-cs) etc...
- Créé à partir de réflexions autour de CommonJS (Node.js), plus orientées client Web
- *EcmaScript Harmony envisage aussi un système de modules qui reprendra les fonctionnalités AMD*

# AMD : Pourquoi ?

## • Pourquoi se contenter de ça en fin de page HTML ?

```
<script src="http://code.jquery.com/jquery-1.7.2.min.js"></script>
<script src="/assets/js/bootstrap/2.0.2/bootstrap-dropdown.js"></script>
<script src="/assets/js/raphael/2.1.0/raphael-min.js"></script>
<script src="/assets/js/mustache/mustache.js"></script>
<script src="/assets/js/tatami/constants.js"></script>
<script src="/assets/js/tatami/standard/tatami.utils.js"></script>
<script src="/assets/js/tatami/standard/tatami.tweets.js"></script>
<script src="/assets/js/tatami/standard/tatami.users.js"></script>
<script src="/assets/js/tatami/standard/tatami.ajax.js"></script>
<script src="/assets/js/tatami/standard/tatami.js"></script>
```

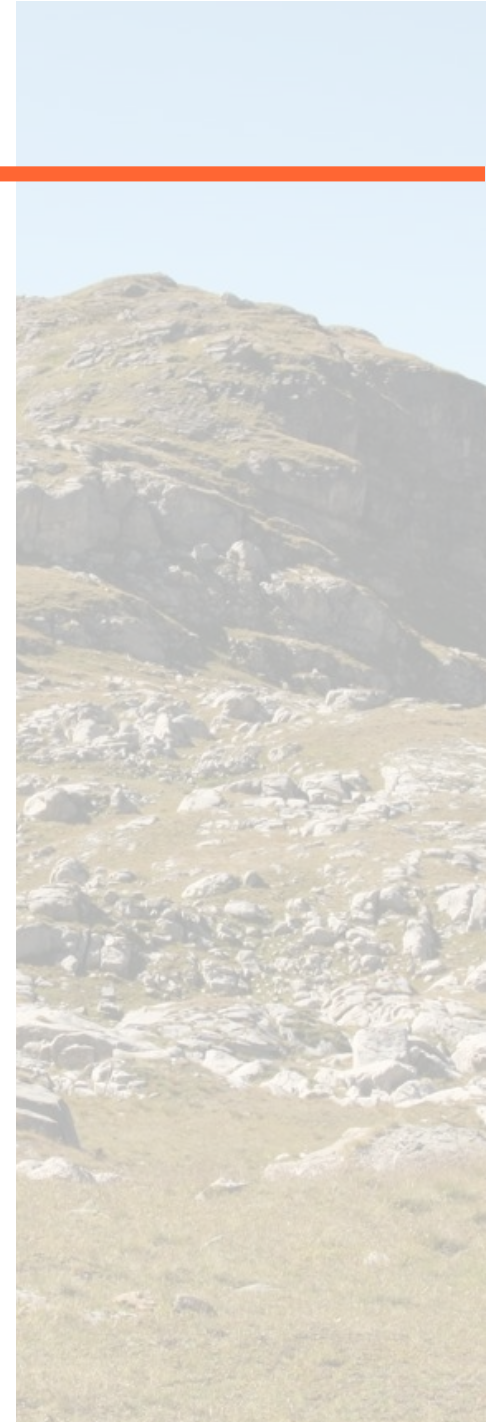
- Pourquoi risquer des conflits de noms de variables ?
- Pourquoi charger des librairies peut-être inutiles ?
- Pourquoi charger plusieurs fois les mêmes librairies ?
- Pourquoi est ce toujours compliqué de faire cohabiter deux versions d'une même librairie ?

# AMD : Pourquoi ?

- Encapsuler du code → Module → Context isolé
- API de création de modules javascript
  - Meilleure organisation du code
  - Gestion des dépendances entre modules
  - Chargement asynchrone via élément natif `<script>` (et non XHR+eval) → efficacité, rapidité
- Dépendances entre modules injectées sous la forme de variables locales → moins de globals

# AMD : Librairies

- La référence : RequireJS (<http://requirejs.org>)
  - James Burke à l'initiative de AMD
  - Complète, plugins, build
- Curl.js → celle que nous utiliserons
  - Plugins, build, fonctionnalités de Deferred
- Lsjs : cache les modules en « local storage »
- Dojo (backdraft)
- NeedJS : librairie légère (<2 ko)
- Brequire (compatible avec CommonJS)
- Inject : librairie créée et utilisée par LinkedIn
- Almond : version lite de RequireJS



# AMD: l'api en 1 slide

- `define` : définit (étonnant non ?) un module
  - Accepte des dépendances - *injectées dans une fonction de callback* → *variables locales*
  - Renvoie LA « valeur » du module (objet, fonction, class, variable...)

```
define(["package2/moduleA", "../moduleB"], function(depA, depB) {  
    //...  
    return function() {  
        //...  
    }  
});
```

- `require` : **idem** `define` mais sans renvoyer de valeur



# AMD : define()

- 1 module = 1 fichier JS (mais outils de *build/optimisation* dispos)
- Organisation idéale par répertoires /« package »
- Ex : définition du module package1/moduleA dans le répertoire /site/package1/moduleA.js (racine : /site)

Non recommandé

Dépendance absolue : /site/package2/moduleA.js

Dépendance relative :  
/site/package1/moduleB.js

```
define("NomModule", ["package2/moduleA", "../moduleB"],  
    function(depA, depB) {  
        //...  
        return function() {  
            //...  
        }  
    }  
);
```



# AMD : define()

- 1 module = 1 fichier JS (mais outils de *build/optimisation* dispos)
- Organisation idéale par répertoires /« package »
- Ex : définition du module package1/moduleA dans le répertoire /site/package1/moduleA.js (racine : /site)

Non recommandé

Dépendance absolue : /site/package2/moduleA.js

Dépendance relative :  
/site/package1/moduleB.js

```
define("NomModule", ["package2/moduleA", "../moduleB"],  
    function(depA, depB) {  
        //...  
        return function() {  
            //...  
        }  
    }  
);
```

# AMD : Configuration

- Nécessité de configurer le loader
  - Les paths, urls, i18n, ...
- Ex : curl.js

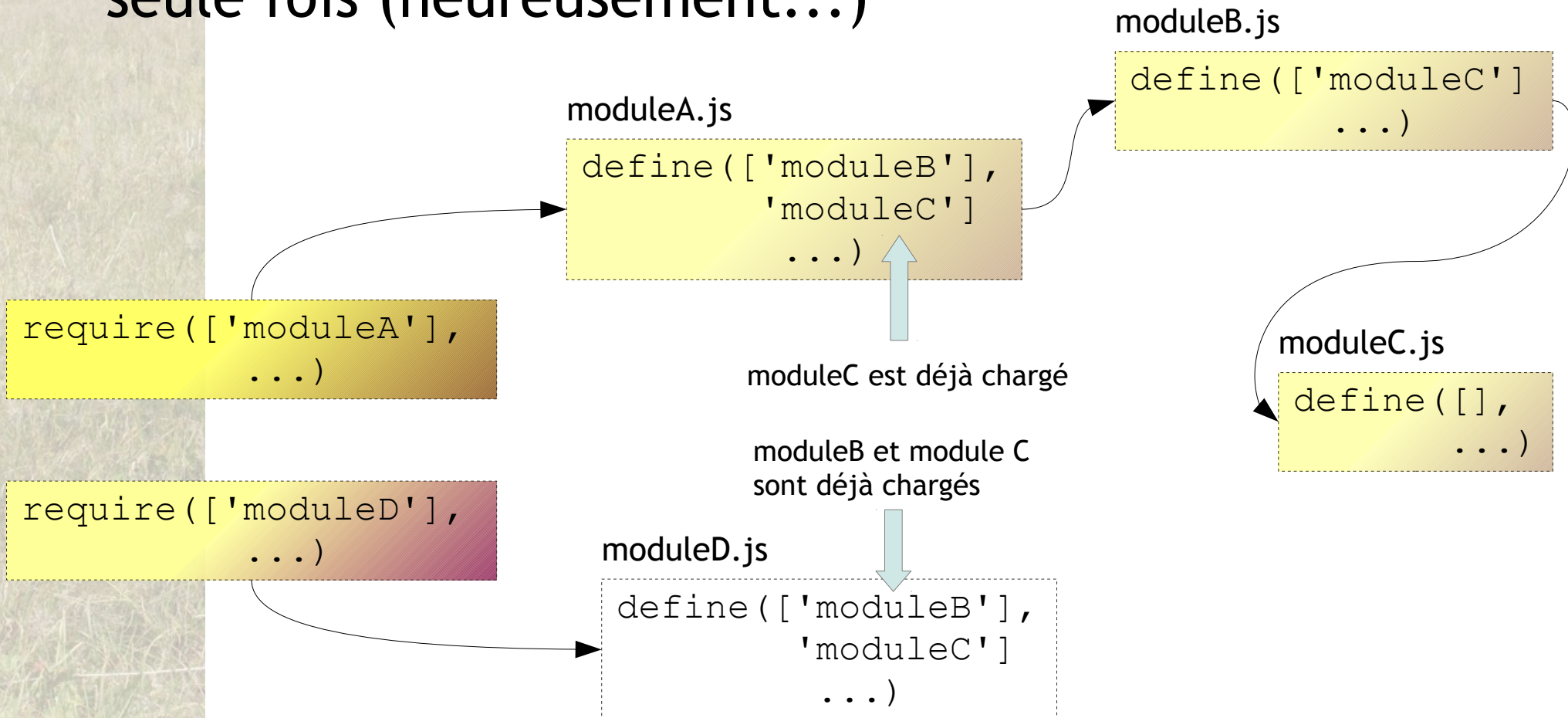
```
curl = {  
  baseUrl:    '/path/to/my/js',  
  pluginPath: 'repertoire/des/plugins/',  
  paths: {  
    curl: '../src/curl',  
    package1: 'projet/package1',  
    my: '../../my-lib/'  
  },  
  locale : 'fr',  
  apiName: 'someOtherName'  
};
```

```
<script src="../src/curl.js"></script>
```



# AMD : Dépendances

- Les dépendances ne sont chargées qu'une seule fois (heureusement...)



- `curl = require` → `curl(...) <=> require(...)`



# AMD: vite, un exemple !

```
//module client/world
define("world");

//module client/hello
define(['./world'], function (txt) {
    return "hello " + txt;
});

//module utils/string
define([], function() {
    return {
        capitalizeFirst : function (str) {
            return str.charAt(0).toUpperCase() + str.slice(1);
        },
        isEmpty : function(str) {
            return /^[s\xa0]*$/ .test(str);
        }
    };
});
```

```
curl(['utils/string', 'client/hello'],
    function (fn, hello) {
        console.log = "Hello world ? > " + fn.capitalizeFirst(hello);
    }
);
```

# AMD: des plugins

- Un plugin est vu comme une dépendance à injecter
- Il remplit un rôle précis
  - Ex : un plugin CSS charge d'abord une feuille de style, peut ensuite normaliser les styles, et enfin les injecter dans la page HTML
- Déclaration de la forme `nom!ressource!param`
  - `js!lib/librairieNonAMD.js`
  - `css!css/styleSheet.css`
  - `text!widget/template.txt`
  - `i18n!nls/messages`
  - **Cas particulier: `domReady!`**



# AMD: exemple de plugins

- Chargement d'une librairie (Mustache)

```
curl(  
  ['client/hello',  
    'js!utils/mustache.js!order!exports=Mustache'  
  ],  
  function (hello, Mustache /*reçoit la variable exportée */) {  
    ...  
  }  
);
```

- Chargement d'une CSS

```
curl(  
  ['client/hello',  
    'css!css/widget.css'  
  ],  
  function (hello) {  
    ...  
  }  
);
```

pas d'injection particulière  
dans la fonction de callback





# AMD: exemple des plugins

- Chargement d'un template (texte)

```
curl(  
    ['client/hello',  
     'text!template/widget.html'  
    ],  
    function (hello, txtWidget ) {  
        ...  
    }  
);
```

- Chargement de messages i18n

```
curl(  
    ['client/hello',  
     'i18n!nls/messages'  
    ],  
    function (hello, messages) {  
        Console.log(messages.accueil + ' :' + hello) ;  
    }  
);
```

# AMD: un peu de cuisine !

- Création d'une « application » qui affiche une timeline Twitter

Lire les 8 derniers tweets de @



- La recette
  - Une page html
  - Un widget Timeline
    - Template Mustache
    - Css propre
    - Librairie de formatage des dates
    - jQuery
  - Pas de chargement inutile

# AMD: un peu de cuisine !

- Création d'une « application » qui affiche une timeline Twitter

Lire les 8 derniers tweets de @



- La recette
  - Une page html
  - Un widget Timeline
    - Template Mustache
    - Css propre
    - Librairie de formatage des dates
    - jQuery
  - Pas de chargement inutile



# AMD: la page HTML (1)

```
<html>...<script>...
  curl = {
    paths: {
      curl : '../src/curl/',
      jquery : 'http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min'
    }
  };
...
  //on s'assure du chargement du DOM et de la console
  curl(['utils/console', 'domReady!'],
    function (console) {
      console.log('Temps de chargement:', new Date() - start);
      document.getElementById("read").onclick = function() {
        //ICI LE CODE POUR AFFICHER NOTRE WIDGET DANS LA DIV tweetsNode
      };
    }
  );
</script>
<body>
<div>
  <label>Lire les 8 derniers tweets de @</label>
  <input type="text" id="user" value="ippontech" />
  <button id="read">Et hop !</button><br>
</div>
<div id="tweetsNode" style="margin-top:20px"></div>
</body>
</html>
```

# AMD: le widget (1)

- Organisé selon cette arborescence
  - widget/twitter/timeline (.js), module principal du widge
  - widget/twitter/css/widget.css, dépendance de type feuille de style
  - widget/twitter/template/template.html, dépendance de type texte représentant le template Mustache
- La CSS

```
/* twitterWidget sera la class de base du widget
*/
.twitterWidget .avatar {
    ...
}
...
```

# AMD: le widget (2)

- Le template Mustache



```
{{#tweets}}  
  <div class="{{baseClass}}">  
    <h3>{{name}}</h3>  
      
    <p>{{&text}}</p>  
    <p class="date">{{date}}</p>  
  </div>  
{{/tweets}}
```

Structure JSON utilisée  
pour alimenter le template

```
{ baseClass: "twitterWidget",  
  tweets : [  
    {  
      'name' : XXXXX,  
      'avatar': XXXXX,  
      'text' : XXXXX,  
      'date' : XXXXX,  
    }, ...  
  ]  
}
```



# AMD: le widget (3)

- Le module de la timeline

```
define(['utils/console','jquery',
  'js!utils/mustache.js!exports=Mustache',
  'text!./template/template.html',
  'css!./css/widget',
  'js!http://stevenlevithan.com/assets/misc/date.format'],
function (console, $, mustache, txtTemplate) {
  //Notre class à renvoyer
  function Timeline(twitterName, nbTweets, container) {
    ...
  }
  //La fonction de rendu de notre widget
  Timeline.prototype.render = function() {
    $.getJSON("http://twitter.com/statuses/user_timeline.json?callback=?",...);
  };

  //la fonction dédiée à l'affichage, "privée"
  Timeline.prototype._display = function(data) {
    var tweets = $.map(data,function(tweet) { ...
      'date': (new Date (tweet.created_at))
        .format("dd/mm/yyyy HH:MM:ss")... });
    //Un peu de travail pour Mustache...
    var view = { baseClass: "twitterWidget", tweets : tweets };
    var output = mustache.render(txtTemplate, view);
    //on affiche le widget avec tous ses tweets
    $(this.container).html(output);
  };

  //Notre class est maintenant définie, on la renvoie
  return Timeline;
});
```

# AMD: la page HTML (2)

- Gestion de l'événement déclenchant le chargement de la timeline


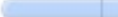




```
...
//on s'assure du chargement du DOM et de la console
curl(['utils/console', 'domReady!'],
    function (console) {
        console.log('Temps de chargement:', new Date() - start);
        document.getElementById("read").onclick = function() {
            curl(['widget/twitter/timeline'],
                function(TimelineTwitter) {
                    var user = document.getElementById("user").value;

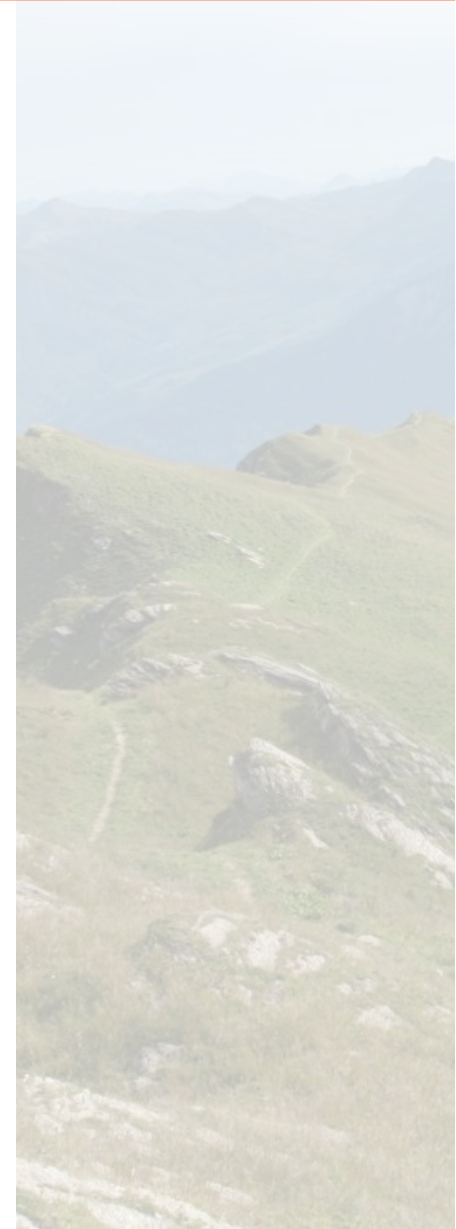
                    //On crée une instance du widget puis on l'affiche
                    var tm = new TimelineTwitter(user, 8,
                        document.getElementById("tweetsNode"));
                    tm.render();
                }
            );
        };
    }
);
...
```

# AMD: Les chargements

## 1) Chargement de la page HTML

- ni jQuery, ni aucun fichier lié au widget

Name Path	Method	Status Text	...	Initiator	Size Content	Time Latency	Timeline
 <b>testTwitter.html</b> /sites/curl/test	GET	304 Not M	t...	Other	232B 1.23KB	15ms 14ms	
 <b>curl.js</b> /sites/curl/src	GET	304 Not M	a...	<u>testTwitter.l</u> Parser	232B 30.08KB	2ms 2ms	
 <b>console.js</b> /sites/curl/test/utlis	GET	304 Not M	a...	<u>curl.js:551</u> Script	231B 401B	2ms 2ms	
 <b>domReady.js</b> /sites/curl/src/curl/plugin	GET	304 Not M	a...	<u>curl.js:551</u> Script	232B 633B	24ms 13ms	
 <b>domReady.js</b> /sites/curl/src/curl	GET	304 Not M	a...	<u>curl.js:551</u> Script	231B 3.38KB	2ms 2ms	


















# AMD: Les chargements

## 2) Affichage de la première timeline

Chargement de jQuery, des plugins AMD, et du widget avec ses dépendances

Appel JSONP Twitter

Name Path	Method	Status Text	...	Initiator	Size Content	Time Latency	Timeline
 domReady.js /sites/curl/src/curl	GET	304 Not M	a...	curl.js:551 Script	2318 3.38KB	2ms 2ms	
 timeline.js /sites/curl/test/widget/twitter	GET	200 OK	a...	curl.js:551 Script	(from cache)	34ms 33ms	
 jquery.min.js ajax.googleapis.com/ajax/libs/jq	GET	200 OK	t...	curl.js:551 Script	(from cache)	0ms 0ms	
 js.js /sites/curl/src/curl/plugin	GET	200 OK	a...	curl.js:551 Script	(from cache)	1ms 0ms	
 text.js /sites/curl/src/curl/plugin	GET	200 OK	a...	curl.js:551 Script	(from cache)	0ms 0ms	
 jquery.color.js /sites/curl/test/utills	GET	200 OK	a...	curl.js:551 Script	(from cache)	0ms 0ms	
 css.js /sites/curl/src/curl/plugin	GET	304 Not M	a...	curl.js:551 Script	(from cache)	0ms 0ms	
 mustache.js /sites/curl/test/utills	GET	304 Not M	a...	js.js:116 Script	(from cache)	0ms 0ms	
 date.format stevenlevithan.com/assets/misc	GET	200 OK	a...	js.js:116 Script	(from cache)	15ms 14ms	
 template.html /sites/curl/test/widget/twitter/te	GET	304 Not M	t...	text.js:54 Script	(from cache)	16ms 0ms	
 widget.css /sites/curl/test/widget/twitter/cs	GET	200 OK	t...	css.js:442 Script	(from cache)	0ms 0ms	
 user_timeline.json twitter.com/statuses	GET	302 Found	t...	jquery.min.j Script	6778 0B	682ms 682ms	
 user_timeline.json twitter.com/statuses	GET	200 OK	a...	http://twitte Redirect	3.08KB 15.77KB	777ms 777ms	
 logo_ippon_normal.png a0.twimg.com/profile_images/45!	GET	200 OK	i...	jquery.min.j Script	(from cache)	0ms 0ms	




# AMD: Les chargements

## 3) Affichage des timelines suivantes

Timeline  
précédente

Appel JSONP  
Twitter

 <b>widget.css</b> /sites/curl/test/widget/twitter/css	GET	200 OK	text/css	CS
 <b>user_timeline.json</b> twitter.com/statuses	GET	302 Found	text/html	ig
 <b>user_timeline.json</b> twitter.com/statuses	GET	200 OK	application/jav...	hi
 <b>logo_ippon_normal.png</b> a0.twimg.com/profile_images/459484608	GET	200 OK	image/png	ig
 <b>user_timeline.json</b> twitter.com/statuses	GET	302 Found	text/html	ig
 <b>user_timeline.json</b> twitter.com/statuses	GET	200 OK	application/jav...	hi
 <b>mootools.twitter_normal.png</b> a0.twimg.com/profile_images/70898977	GET	200 OK	image/png	ig

# AMD: Les performances

- Le risque : beaucoup de petits modules (des centaines...) → autant de requêtes HTTP !!
  - Système de build et d'optimisations
    - regroupement de plusieurs modules dans un seul fichier
    - Compilation (ex : Google Closure Compiler)
- Performances de chargement bien meilleures que par utilisation de XHR + eval (~ 50%)
- Charger les modules uniquement quand nécessaire
  - Possibilité d'exécuter un `require` dans un `define` → aussi la solution pour éviter les références circulaires

# AMD: Les + de Curl

- Chaque loader AMD peut apporter quelques fonctionnalités spécifiques
- Ex de curl.js : Deferred, chainage etc...

```
curl(['js!has.js'])
  .next(['model/client', 'utils/string', 'utils/console'],
    function (client, fn, console) {
      // on prépare les données...
    }
  )
  .next(['widget/twitter/timeline', 'domReady!'])
  .then(
    function (Timeline) {
      // tout est ok, Timeline chargée et DOM prêt
    },
    function (ex) {
      // en cas de problème, code exécuté
    }
  );
```



# Questions ?