Browserify

Browserify est une bibliothèque javascript, utilisable en ligne de commande.
Browserify permet le découpage d'une application javascript en modules :

- Par la création d'un seul point d'entrée dans l'application en façade (**bundle.js**)
- En facilitant la référence à des fichiers annexes (**require(“./autreFichier”)**)

Sans Browserify

```
<!DOCTYPE html>
<html data-ng-app="App">
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
    <script
src="libs/angular.min.js"></script>
    <script src="libs/angular-
route.min.js"></script>

    <script src="js/app2.js"></script>
    <script
src="js/controller2.js"></script>
    <script src="js/app.js"></script>
    <script
src="js/controller1.js"></script>
    <script
src="js/directives.js"></script>
    <script src="js/routes.js"></script>
    ...

</head>
```

Avec Browserify

```
<!DOCTYPE html>
<html data-ng-app="App">
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
    <script
src="libs/angular.min.js"></script>
    <script src="libs/angular-
route.min.js"></script>

    <script src="js/bundle.js"></script>

</head>
```

-- Installation

1. Télécharger et installer **Node.js**
2. Démarrer ensuite l'invite de commande node
3. Frappez et exécuter :

```
npm install -g browserify
```

-- Mise en oeuvre

-- Projet simple sans Browserify

L'application affiche simplement 2 pages (définies dans le fichier **routes.js**), à partir de contrôleurs différents.

Le module principal est **App** (défini dans **app.js**), il requiert le module **App2**, le contrôleur **App1Controller**, la directive **dirClient**

Pour que l'application fonctionne correctement, tous les fichiers **js** doivent être inclus dans le fichier HTML, et dans un ordre cohérent, respectant les dépendances (app2.js avant app.js).

```
<!DOCTYPE html>
<html data-ng-app="App">
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
    <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.11/angular.min.js"></script
>
    <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.11/angular-route.min.js"></
script>

    <script src="js/app2.js"></script>
    <script src="js/controller2.js"></script>
    <script src="js/app.js"></script>
    <script src="js/controller1.js"></script>
    <script src="js/directives.js"></script>
    <script src="js/routes.js"></script>

</head>
<body>
    <h1>Main</h1>
    <ul>
        <li><a href="#page1">Page 1</a></li>
        <li><a href="#page2">Page 2</a></li>
    </ul>
    <data-ng-view></data-ng-view>
</body>
</html>
```

```
angular.module("App2", []);
```

```
angular.module("App2").controller("App2Controller", ["$scope",function($scope) {
    $scope.client="Durand";
}]);
```

```
angular.module("App", ['ngRoute','App2']);
```

```
angular.module("App").controller("AppController", ["$scope",function($scope) {
```

```
    $scope.client="Dupond";
}]);
```

```
angular.module("App").directive("dirClient", function() {
    return {
        template : "<div>Client : {{client}}</div>"
    };
})
```

```
angular.module("App").config(["$routeProvider",function($routeProvider) {
    $routeProvider.when('/page1', {
        controller: 'AppController',
        templateUrl: 'templates/main.html'
    }).when('/page2', {
        controller: 'App2Controller',
        templateUrl: 'templates/main.html'
    }).otherwise({
        redirectTo: '/page1'
    });
}]);
```

```
<dir-client></dir-client>
```

-- Avec Browserify

-- Refactorisation du code

Le fichier de l'application (app.js) déclare toutes ses composantes, et utilise **require(“./fileName”)** pour référencer une dépendance à un fichier **filename.js** se trouvant dans le même dossier (**./**).

```
angular.module("App", ['ngRoute',require("./app2")]);
angular.module("App").controller("App1Controller",
["$scope",require("./controller1")]);
angular.module("App").directive("dirClient",require("./directives"));
angular.module("App").config(["$routeProvider",require("./routes")]);
```

Le module **App2** utilise le même principe, et exporte (**module.exports**) son nom (**name**), afin qu'il soit utilisé par **App1** comme dépendance :

```
angular.module("App2", []);
angular.module("App2").controller("App2Controller",
["$scope",require("./controller2")]);
module.exports=angular.module("App2").name;
```

```
module.exports=function($scope) {
    $scope.client="Durand";
};
```

Chaque composant (contrôleur, directive...) utilisé est défini dans son propre fichier et exporté en utilisant ce même principe (**module.exports**) :

```
module.exports=function($scope) {
    $scope.client="Dupond";
};
```

```
module.exports=function() {
    return {
        template : "<div>Client : {{client}}</div>"
    };
};
```

```
module.exports=function($routeProvider) {
    $routeProvider.when('/page1', {
        controller: 'AppController',
        templateUrl: 'templates/main.html'
    }).when('/page2', {
        controller: 'App2Controller',
        templateUrl: 'templates/main.html'
    }).otherwise({
        redirectTo: '/page1'
    });
};
```

Le fichier template main.html est inchangé

-- Création du Bundle

Pour créer le bundle Browserify, qui va réunir l'ensemble des sources JS définies dans l'application :

1. Repasser à l'invite de commandes Node.js
2. Aller dans le dossier js du projet
3. Entrer la commande :

```
browserify app.js -o bundle.js
```

Corrigez les erreurs en cas d'echec de la génération.
En cas de succès, vérifiez dans le dossier la génération du fichier **bundle.js** (raffraîchissez le dossier contenant le fichier dans Eclipse)

-- Intégration du Bundle

```
<!DOCTYPE html>
<html data-ng-app="App">
<head>
<meta charset="UTF-8">
<title>Insert title here</title>
    <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.11/angular.min.js"></script
>
    <script
src="http://ajax.googleapis.com/ajax/libs/angularjs/1.3.11/angular-route.min.js"></
script>

    <script src="js/bundle.js"></script>

</head>
...
```

-- Intégration d'angular

Il est également possible d'utiliser Browserify pour l'intégration d'angularJS, et de ses modules liés (ngCookies, ngRoute...). Mais c'est une méthode qui ne permet pas d'utiliser les CDN (google API par exemple).

-- Minification

Utilisez un des outils en ligne, du genre http://jscompress.com/ , http://javascript-minifier.com/ ...

Résultat :

```
!function e(r,t,n){function o(i,p){if(!t[i]){if(!r[i]){var u="function"==typeof
require&&require;if(!p&&u)return u(i,!0);if(l)return l(i,!0);var c=new
Error("Cannot find module '"+i+"'");throw c.code="MODULE_NOT_FOUND",c}var
a=t[i]={exports:{}};r[i][0].call(a.exports,function(e){var t=r[i][1][e];return
o(t?t:e)},a,a.exports,e,r,t,n)}return t[i].exports}for(var l="function"==typeof
require&&require,i=0;i<n.length;i++)o(n[i]);return
o}({1:[function(e){angular.module("App",["ngRoute",e("./app2")]).controller("App1Co
ntroller",["$scope",e("./controller1")]).directive("dirClient",e("./directives")).c
onfig(["$routeProvider",e("./routes")])},{"./app2":2,"./controller1":3,"./directive
s":5,"./routes":6}],2:[function(e,r){angular.module("App2",[]).controller("App2Cont
roller",["$scope",e("./controller2")]),r.exports=angular.module("App2").name},{"./c
ontroller2":4}],3:[function(e,r){r.exports=function(e){e.client="Dupond"}},{}],4:[f
unction(e,r){var
```

```
t=function(e){e.client="Durand"};r.exports=t},{}],5:[function(e,r){r.exports=functi
on(){return{template:"<div>Client :
{{client}}</div>"}}},{}],6:[function(e,r){r.exports=function(e){e.when("/page1",{co
ntroller:"App1Controller",templateUrl:"templates/main.html"}).when("/page2",{contro
ller:"App2Controller",templateUrl:"templates/main.html"}).otherwise({redirectTo:"/p
age1"})}},{}]},{},[1]);
```